APPENDIX

Package Index

Other Packages

- package .default
- package ecardfile.appl
- package ecardfile.dbappl
- package multiserv.applservlet
- package multiserv.dbmgr
- package multiserv.sessionmgr
- package multiserv.util

package.default

Class Index

- LoginServlet
- SearchServlet

Class LoginServlet

```
Object
   |
   +---GenericServlet
           |
           +---HttpServlet
                   |
                   +---ApplServlet
                           |
                           +---LoginServlet
```

public class **LoginServlet**
extends ApplServlet
Used to override the getApplicationInterface method defined in ApplServlet. This method provides the link between the generic components of package multiserv.applservlet and the application specific functionality.

Constructor Index

•LoginServlet()

Method Index

•getApplicationInterface()
Overrides the getApplicationInterface method defined in class ApplServlet.
•init(ServletConfig)
Called when the servlet first gets loaded.

Constructors

●**LoginServlet**

```
public LoginServlet()
```

Methods

●**getApplicationInterface**

```
public multiserv.applservlet.ApplicationInterface
getApplicationInterface()
```

Overrides the getApplicationInterface method defined in class ApplServlet. This method must return an instance of the application specific class which implements ApplicationInterface.
Overrides:
getApplicationInterface in class ApplServlet

init

```
public void init(ServletConfig config) throws ServletException
```

Called when the servlet first gets loaded. Do our application specific servlet initialization here after calling init() in ApplServlet.

Parameters:

config - servlet configuration information.

Throws: ServletException

a problem occurred during the initialization of the servlet.

Overrides:

init in class ApplServlet

Class SearchServlet

```
Object
   |
   +---GenericServlet
           |
           +---HttpServlet
                   |
                   +---ApplServlet
                           |
                           +---SearchServlet
```

public class **SearchServlet**
extends ApplServlet
Used to override the getApplicationInterface method defined in ApplServlet. This method provides the link between the generic components of package multiserv.applservlet and the application specific functionality.

Constructor Index

- SearchServlet()

Method Index

- getApplicationInterface()
  Overrides the getApplicationInterface method defined in class ApplServlet.
- init(ServletConfig)
  Called when the servlet first gets loaded.

Constructors

- **SearchServlet**

```
public SearchServlet()
```

Methods

- **getApplicationInterface**

```
public multiserv.applservlet.ApplicationInterface
getApplicationInterface()
```

Overrides the getApplicationInterface method defined in class ApplServlet. This method must return an instance of the application specific class which implements ApplicationInterface.
Overrides:

getApplicationInterface in class ApplServlet

init

```
public void init(ServletConfig config) throws ServletException
```

Called when the servlet first gets loaded. Do our application specific servlet initialization here after calling init() in ApplServlet.

Parameters:

config - servlet configuration information.

Throws: ServletException

a problem occurred during the initialization of the servlet.

Overrides:

init in class ApplServlet

package ecardfile.appl

Interface Index

- CommonConfig

Class Index

- CommonApplicationInterface
- EcardNotifier
- LoginApplicationInterface
- SearchApplicationInterface

Interface ecardfile.appl.CommonConfig

public abstract interface **CommonConfig**
Defines the common constants used by the application

Variable Index

- ADDLIST_TAG
- ADDRESSID_TAG
- ADDUSERCONFIRM_TAG
- ADDUSER_TAG
- ALTFIRSTNAME_COL
- ALTFIRSTNAME_TAG
- ALTFIRSTNAME_TOKEN
- BANNER_TOKEN
- BUSINESSCOMMENT_TOKEN
- BUTTON_TAG
- CANCEL_TAG
- CARDEXTRA
- CARDID_TAG
- CARDID_TOKEN
- CATEGORY_TOKEN
- CHANGE
- CHANGEDETAILS_TAG
- CHANGEWHEREAMI_TAG
- COMPANYNAME_COL
- COMPANYNAME_ENC_TOKEN
- COMPANYNAME_TAG
- COMPANYNAME_TOKEN
- CONFIRM_TAG
- CONTACT_COL
- CREATEID_TAG
- DATEOFENTRY_TAG
- DATEOFENTRY_TOKEN
- DELETEUSERCONFIRM_TAG
- DELETEUSER_TAG
- DELETE_TAG
- DISPLAY
- DISPLAYFMT_TAG
- DISPLAYFMT_TOKEN
- DISPLAYLIST_TAG
- DISPLAY_PAGE_TAG

- DOADDWHERE_TAG
- DOCHANGEDETAILS_TAG
- DOCHANGEWHERE_TAG
- DODOWNLOADCARD_TAG
- DODOWNLOADPLIST_TAG
- DODOWNLOADSINGLE_TAG
- DOSEARCH_TAG
- DOUPDATEPLIST_TAG
- DOWNLOADFMT_TAG
- DOWNLOADID_TAG
- DOWNLOADSINGLE_TAG
- DOWNLOAD_TAG
- ECARDID_TAG
- ECARDID_TOKEN
- EDITPRIVACY_TAG
- EMAILAUTH_COL
- EMAILAUTH_TAG
- EMAILAUTH_TOKEN
- EMAILID_TAG
- EPASSWORDCONF_TAG
- EPASSWORD_TAG
- EXPIRYDATE_COL
- FIND_PASSWORD_TAG
- FIRSTNAME_COL
- FIRSTNAME_ENC_TOKEN
- FIRSTNAME_TAG
- FIRSTNAME_TOKEN
- FMT_TAG
- FULLNAME_TOKEN
- HTML
- ID_REWRITE_TOKEN
- ID_TOKEN
- JOBTITLE_TOKEN
- LASTNAME_COL
- LASTNAME_ENC_TOKEN
- LASTNAME_TAG
- LASTNAME_TOKEN
- LC_PRIVACYPREFIX
- LISTITEMS_TOKEN
- LOGIN_SERVLET_TOKEN
- MASK_COL
- MASK_TOKEN
- MIDDLENAME_COL
- MIDDLENAME_TAG
- MIDDLENAME_TOKEN

- MSG_LIST_TOKEN
- NEWUSER_TAG
- OK_TAG
- ONETIME_TAG
- PAGE_TAG
- PAGE_TOKEN
- PASSWORD_COL
- PASSWORD_TOKEN
- PDAPAGE_TAG
- PERSONALLISTID_COL
- PERSONALLISTITEMS_TOKEN
- PHONEID_TAG
- PRIVACYPREFIX
- PVTALTFIRSTNAME_TOKEN
- PVTBUSINESSCOMMENT_TOKEN
- PVTCOMPANYNAME_TOKEN
- PVTCONTACT_COL
- PVTFIRSTNAME_TOKEN
- PVTJOBTITLE_TOKEN
- PVTLASTNAME_TOKEN
- PVTLISTID_COL
- PVTLISTID_TOKEN
- PVTMIDDLENAME_TOKEN
- PVTSUFFIX_TOKEN
- PVTTITLE_TOKEN
- PVTWEBPAGEURL_TOKEN
- ROWID_TAG
- SEARCHID_TAG
- SEARCHLISTITEMS_TOKEN
- SEARCHNAME_TAG
- SEARCH_SERVLET_TOKEN
- SEARCH_TAG
- SITE_ADDRESS_TOKEN
- SOUNDEX_TAG
- SUFFIX_COL
- SUFFIX_TOKEN
- TEMPLATE
- TITLE_COL
- TITLE_TOKEN
- UPDATE_TAG
- USERINFOID_TAG
- USERSID_COL
- USERSID_TOKEN
- WEBPAGEURL_ENC_TOKEN
- WEBPAGEURL_TOKEN

- WHEREAMI
- WHEREAMI_TAG
- WML
- monthNames

  Static array with the names of the months for formatting dates

Variables

- ADDLIST_TAG

```
public static final java.lang.String ADDLIST_TAG
```

- ADDRESSID_TAG

```
public static final java.lang.String ADDRESSID_TAG
```

- ADDUSERCONFIRM_TAG

```
public static final java.lang.String
ADDUSERCONFIRM_TAG
```

- ADDUSER_TAG

```
public static final java.lang.String ADDUSER_TAG
```

- ALTFIRSTNAME_COL

```
public static final java.lang.String ALTFIRSTNAME_COL
```

- ALTFIRSTNAME_TAG

```
public static.final java.lang.String ALTFIRSTNAME_TAG
```

- ALTFIRSTNAME_TOKEN

```
public static final java.lang.String
ALTFIRSTNAME_TOKEN
```

- BANNER_TOKEN

```
public static final java.lang.String BANNER_TOKEN
```

- BUSINESSCOMMENT_TOKEN

```
public static final java.lang.String
BUSINESSCOMMENT_TOKEN
```

- BUTTON_TAG

```
public static final java.lang.String BUTTON_TAG
```

- CANCEL_TAG

```
public static final java.lang.String CANCEL_TAG
```

CARDEXTRA

public static final short CARDEXTRA
CARDID_TAG

public static final java.lang.String CARDID_TAG
●CARDID_TOKEN

public static final java.lang.String CARDID_TOKEN
●CATEGORY_TOKEN

public static final java.lang.String CATEGORY_TOKEN
●CHANGE

public static final short CHANGE
●CHANGEDETAILS_TAG

public static final java.lang.String
CHANGEDETAILS_TAG
●CHANGEWHEREAMI_TAG

public static final java.lang.String
CHANGEWHEREAMI_TAG
●COMPANYNAME_COL

public static final java.lang.String COMPANYNAME_COL
●COMPANYNAME_ENC_TOKEN

public static final java.lang.String
COMPANYNAME_ENC_TOKEN
●COMPANYNAME_TAG

public static final java.lang.String COMPANYNAME_TAG
●COMPANYNAME_TOKEN

public static final java.lang.String
COMPANYNAME_TOKEN
●CONFIRM_TAG

public static final java.lang.String CONFIRM_TAG
●CONTACT_COL

public static final java.lang.String CONTACT_COL
●CREATEID_TAG

public static final java.lang.String CREATEID_TAG
●DATEOFENTRY_TAG

public static final java.lang.String DATEOFENTRY_TAG
DATEOFENTRY_TOKEN

public static final java.lang.String
DATEOFENTRY_TOKEN
- DELETEUSERCONFIRM_TAG

public static final java.lang.String
DELETEUSERCONFIRM_TAG
- DELETEUSER_TAG

public static final java.lang.String DELETEUSER_TAG
- DELETE_TAG

public static final java.lang.String DELETE_TAG
- DISPLAY

public static final short DISPLAY
- DISPLAYFMT_TAG

public static final java.lang.String DISPLAYFMT_TAG
- DISPLAYFMT_TOKEN

public static final java.lang.String DISPLAYFMT_TOKEN
- DISPLAYLIST_TAG

public static final java.lang.String DISPLAYLIST_TAG
- DISPLAY_PAGE_TAG

public static final java.lang.String DISPLAY_PAGE_TAG
- DOADDWHERE_TAG

public static final java.lang.String DOADDWHERE_TAG
- DOCHANGEDETAILS_TAG

public static final java.lang.String
DOCHANGEDETAILS_TAG
- DOCHANGEWHERE_TAG

public static final java.lang.String
DOCHANGEWHERE_TAG
- DODOWNLOADCARD_TAG

public static final java.lang.String
DODOWNLOADCARD_TAG
DODOWNLOADPLIST_TAG

public static final java.lang.String
DODOWNLOADPLIST_TAG
●DODOWNLOADSINGLE_TAG

public static final java.lang.String
DODOWNLOADSINGLE_TAG
●DOSEARCH_TAG

public static final java.lang.String DOSEARCH_TAG
●DOUPDATEPLIST_TAG

public static final java.lang.String
DOUPDATEPLIST_TAG
●DOWNLOADFMT_TAG

public static final java.lang.String DOWNLOADFMT_TAG
●DOWNLOADID_TAG

public static final java.lang.String DOWNLOADID_TAG
●DOWNLOADSINGLE_TAG

public static final java.lang.String
DOWNLOADSINGLE_TAG
●DOWNLOAD_TAG

public static final java.lang.String DOWNLOAD_TAG
●ECARDID_TAG

public static final java.lang.String ECARDID_TAG
●ECARDID_TOKEN

public static final java.lang.String ECARDID_TOKEN
●EDITPRIVACY_TAG

public static final java.lang.String EDITPRIVACY_TAG
●EMAILAUTH_COL

public static final java.lang.String EMAILAUTH_COL
●EMAILAUTH_TAG

public static final java.lang.String EMAILAUTH_TAG

●EMAILAUTH_TOKEN

public static final java.lang.String EMAILAUTH_TOKEN
●EMAILID_TAG

public static final java.lang.String EMAILID_TAG
●EPASSWORDCONF_TAG

public static final java.lang.String
EPASSWORDCONF_TAG
●EPASSWORD_TAG

public static final java.lang.String EPASSWORD_TAG
●EXPIRYDATE_COL

public static final java.lang.String EXPIRYDATE_COL
●FIND_PASSWORD_TAG

public static final java.lang.String
FIND_PASSWORD_TAG
●FIRSTNAME_COL

public static final java.lang.String FIRSTNAME_COL
●FIRSTNAME_ENC_TOKEN

public static final java.lang.String
FIRSTNAME_ENC_TOKEN
●FIRSTNAME_TAG

public static final java.lang.String FIRSTNAME_TAG
●FIRSTNAME_TOKEN

public static final java.lang.String FIRSTNAME_TOKEN
●FMT_TAG

public static final java.lang.String FMT_TAG
●FULLNAME_TOKEN

public static final java.lang.String FULLNAME_TOKEN
●HTML

public static final java.lang.String HTML
●ID_REWRITE_TOKEN

public static final java.lang.String ID_REWRITE_TOKEN

ID_TOKEN

public static final java.lang.String ID_TOKEN
●JOBTITLE_TOKEN

public static final java.lang.String JOBTITLE_TOKEN
●LASTNAME_COL

public static final java.lang.String LASTNAME_COL
●LASTNAME_ENC_TOKEN

public static final java.lang.String
LASTNAME_ENC_TOKEN
●LASTNAME_TAG

public static final java.lang.String LASTNAME_TAG
●LASTNAME_TOKEN

public static final java.lang.String LASTNAME_TOKEN
●LC_PRIVACYPREFIX

public static final java.lang.String LC_PRIVACYPREFIX
●LISTITEMS_TOKEN

public static final java.lang.String LISTITEMS_TOKEN
●LOGIN_SERVLET_TOKEN

public static final java.lang.String
LOGIN_SERVLET_TOKEN
●MASK_COL

public static final java.lang.String MASK_COL
●MASK_TOKEN

public static final java.lang.String MASK_TOKEN
●MIDDLENAME_COL

public static final java.lang.String MIDDLENAME_COL
●MIDDLENAME_TAG

public static final java.lang.String MIDDLENAME_TAG
●MIDDLENAME_TOKEN

public static final java.lang.String MIDDLENAME_TOKEN
MSG_LIST_TOKEN

public static final java.lang.String MSG_LIST_TOKEN
NEWUSER_TAG

public static final java.lang.String NEWUSER_TAG
●OK_TAG

public static final java.lang.String OK_TAG
●ONETIME_TAG

public static final java.lang.String ONETIME_TAG
●PAGE_TAG

public static final java.lang.String PAGE_TAG
●PAGE_TOKEN

public static final java.lang.String PAGE_TOKEN
●PASSWORD_COL

public static final java.lang.String PASSWORD_COL
●PASSWORD_TOKEN

public static final java.lang.String PASSWORD_TOKEN
●PDAPAGE_TAG

public static final java.lang.String PDAPAGE_TAG
●PERSONALLISTID_COL

public static final java.lang.String
PERSONALLISTID_COL
●PERSONALLISTITEMS_TOKEN

public static final java.lang.String
PERSONALLISTITEMS_TOKEN
●PHONEID_TAG

public static final java.lang.String PHONEID_TAG
●PRIVACYPREFIX

public static final java.lang.String PRIVACYPREFIX
●PVTALTFIRSTNAME_TOKEN

public static final java.lang.String
PVTALTFIRSTNAME_TOKEN
PVTBUSINESSCOMMENT_TOKEN

public static final java.lang.String
PVTBUSINESSCOMMENT_TOKEN

- PVTCOMPANYNAME_TOKEN

public static final java.lang.String
PVTCOMPANYNAME_TOKEN

- PVTCONTACT_COL

public static final java.lang.String PVTCONTACT_COL

- PVTFIRSTNAME_TOKEN

public static final java.lang.String
PVTFIRSTNAME_TOKEN

- PVTJOBTITLE_TOKEN

public static final java.lang.String
PVTJOBTITLE_TOKEN

- PVTLASTNAME_TOKEN

public static final java.lang.String
PVTLASTNAME_TOKEN

- PVTLISTID_COL

public static final java.lang.String PVTLISTID_COL

- PVTLISTID_TOKEN

public static final java.lang.String PVTLISTID_TOKEN

- PVTMIDDLENAME_TOKEN

public static final java.lang.String
PVTMIDDLENAME_TOKEN

- PVTSUFFIX_TOKEN

public static final java.lang.String PVTSUFFIX_TOKEN

- PVTTITLE_TOKEN

public static final java.lang.String PVTTITLE_TOKEN

- PVTWEBPAGEURL_TOKEN

public static final java.lang.String
PVTWEBPAGEURL_TOKEN

- ROWID_TAG

public static final java.lang.String ROWID_TAG

SEARCHID_TAG

public static final java.lang.String SEARCHID_TAG

- SEARCHLISTITEMS_TOKEN

public static final java.lang.String SEARCHLISTITEMS_TOKEN

- SEARCHNAME_TAG

public static final java.lang.String SEARCHNAME_TAG

- SEARCH_SERVLET_TOKEN

public static final java.lang.String SEARCH_SERVLET_TOKEN

- SEARCH_TAG

public static final java.lang.String SEARCH_TAG

- SITE_ADDRESS_TOKEN

public static final java.lang.String SITE_ADDRESS_TOKEN

- SOUNDEX_TAG

public static final java.lang.String SOUNDEX_TAG

- SUFFIX_COL

public static final java.lang.String SUFFIX_COL

- SUFFIX_TOKEN

public static final java.lang.String SUFFIX_TOKEN

- TEMPLATE

public static final java.lang.String TEMPLATE

- TITLE_COL

public static final java.lang.String TITLE_COL

- TITLE_TOKEN

public static final java.lang.String TITLE_TOKEN

- UPDATE_TAG

public static final java.lang.String UPDATE_TAG

- USERINFOID_TAG

public static final java.lang.String USERINFOID_TAG

USERSID_COL

public static final java.lang.String USERSID_COL

- USERSID_TOKEN

public static final java.lang.String USERSID_TOKEN

- WEBPAGEURL_ENC_TOKEN

public static final java.lang.String WEBPAGEURL_ENC_TOKEN

- WEBPAGEURL_TOKEN

public static final java.lang.String WEBPAGEURL_TOKEN

- WHEREAMI

public static final short WHEREAMI

- WHEREAMI_TAG

public static final java.lang.String WHEREAMI_TAG

- WML

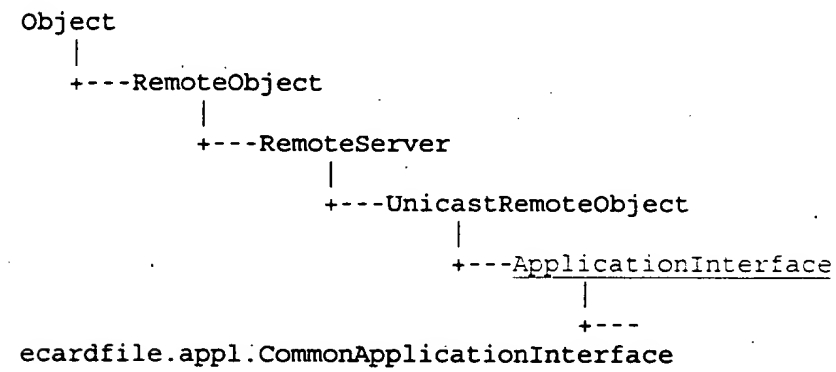
public static final java.lang.String WML

- monthNames

public static final java.lang.String[] monthNames

Static array with the names of the months for formatting dates

Class ecardfile.appl.CommonApplicationInterface



public abstract class **CommonApplicationInterface**
extends ApplicationInterface
implements SessionTags, CommonConfig

Version:

$Id: CommonApplicationInterface.java,v 1.44 1999/12/03 02:05:45 peter Exp $

See Also:

ApplicationInterface, SearchApplicationInterface, LoginApplicationInterface, RequestHandler, LoginServlet, SearchServlet

Variable Index

- defaultPdaPage
- verboseErrors

Constructor Index

- ecardfile.appl.CommonApplicationInterface()

Method Index

- accessDenied(String, HttpServletRequest, HttpServletResponse)
  Called when a received request does not contain a valid session id.
- addBannerToken(Hashtable)
  Add the banner text to the token table
- checkAccess(HttpServletRequest)
  Implementation of ApplicationInterface::checkAccess().
- checkPrivacyAccess(DatabaseConnection2, long, long, Hashtable)
- db_error(HttpServletRequest, HttpServletResponse)
  A database error has occurred

- db_error(HttpServletRequest, HttpServletResponse, String)
  A database error has occurred
- destroy()
  Brush teeth before going to bed.
- doc_access_error(HttpServletRequest, HttpServletResponse)
  A document access error has occurred
- getCookie(HttpServletRequest)
  Extract and return our cookie contents from the original HTTP request.
- getCookieTag()
- getFullPath(String)
  Given an HTML document name determine if a full path was specified.
- getOperation(HttpServletRequest)
  Determine the type of operation to be invoked by the request.
- getPassword(HttpServletRequest)
  Get a string representing the user's password
- getServletImgBtnParameter(HttpServletRequest)
- getSessionId(HttpServletRequest)
  Extract and return the session id from the original HTTP request.
- getUserId(HttpServletRequest)
  Get a string representing the user identification
- hiddenField(String, String)
  Return a string of HTML specifying a hidden field.
- init(ApplServlet, String, String)
  Used to initialize the application specific class which implements this interface.
- io_error(HttpServletRequest, HttpServletResponse)
  An io exception has occurred
- isLoggedIn(String, Session)
  Called to check if a user is logged in
- nfe_error(HttpServletRequest, HttpServletResponse)
  A NumberFormatException has occurred
- nfe_error(HttpServletRequest, HttpServletResponse, String)
  A NumberFormatException has occurred
- nse_error(HttpServletRequest, HttpServletResponse)
  A non-serializable exception has occurred
- operationRequiresLogin(String)
  Called to check if a specified operation requires user login.
- re_error(HttpServletRequest, HttpServletResponse)
  A Remote Exception has occurred
- sae_error(HttpServletRequest, HttpServletResponse)
  A Session Access Exception has occurred
- sendDocument(String, HttpServletResponse)
  Send an HTML document to the user replacing the predefined character sequences with their associated values.
- sendError(String, String, String, HttpServletResponse)

- sendError(Hashtable, String, String, String, HttpServletResponse)
- sendError(Hashtable, String, String, HttpServletResponse)
- sendError(HttpServletRequest, String, String, HttpServletResponse)
  General error handling routine.
- sendLoginScreen(HttpServletRequest, HttpServletResponse, String)
- sendLoginScreen(HttpServletRequest, HttpServletResponse, String, Hashtable)
- sendMessage(String, String, Hashtable, String, HttpServletResponse)
- sendMessage(String, String, HttpServletResponse)
- sendMessage(String, Hashtable, String, HttpServletResponse)
- sendParseDocument(Hashtable, String, String, HttpServletResponse)
- sendParseDocument(Hashtable, String, HttpServletResponse)
  Send an HTML document to the user replacing the predefined character sequences with their associated values.
- sendParseTextFile(String, String, Hashtable, String, String, HttpServletResponse)
- sendParseTextFile(String, Hashtable, String, String, HttpServletResponse)
- sendParseTextFile(Hashtable, String, String, HttpServletResponse)
  Send a text file to the user replacing the predefined character sequences with their associated values.
- sendSearchScreen(HttpServletRequest, HttpServletResponse)
- sendSearchScreen(HttpServletRequest, HttpServletResponse, String)
- sendSearchScreen(HttpServletRequest, HttpServletResponse, String, Hashtable)
- sessionFailure(String, HttpServletRequest)
  Implementation of ApplicationInterface:sessionFailure Notify the application that an attempt to access a sesssion using sessionId failed.
- unknownOperation(HttpServletRequest, HttpServletResponse)
  An unknown operation has been requested
- validateSession(String, Session, HttpServletRequest)
  Validate the passed session This method can also be used to implement any security checks or other checks such as if undesirables are accessing the site - such as SPAMMERS
- verboseError(String)
  Method, which displays verbose error, messages to user if debug is turned on

Variables

- **defaultPdaPage**

```
public static final java.lang.String defaultPdaPage
```

- **verboseErrors**

```
protected boolean verboseErrors
```

Constructors

- **CommonApplicationInterface**

```
public CommonApplicationInterface() throws RemoteException
```

Methods

●accessDenied

```
public void accessDenied(String operation,
                         HttpServletRequest req,
                         HttpServletResponse resp)
```

Called when a received request does not contain a valid session id. The application should return an error/warning document to the user. If the error occurred during a normal operation (i.e. other than Logging in or Logging out) it was more than likely due to the session having expired. In this case just bring up the login screen.

Parameters:

operation - The operation which was being attempted.

req - The original HTTP request.

resp - The HTTP response

Overrides:

accessDenied in class ApplicationInterface

●addBannerToken

```
protected void addBannerToken(Hashtable tokens)
```

Add the banner text to the token table

Parameters:

tokens - the token table

●checkAccess

```
public boolean checkAccess(HttpServletRequest req)
```

Implementation of ApplicationInterface::checkAccess(). Check the HTTP request data and decide if access should be allowed. We check to see if the IP Address is locked out.

Parameters:

req - The original HTTP request data.

Returns:

An indication if access is to be granted

Overrides:

checkAccess in class ApplicationInterface

●checkPrivacyAccess

```
public short checkPrivacyAccess(DatabaseConnection2 jdbc,
                                long userId,
                                long cardId,
                                Hashtable cardRow)
```

●db_error

```
public void db_error(HttpServletRequest req,
                     HttpServletResponse resp)
```

A database error has occurred

Overrides:

db_error in class ApplicationInterface

- **db_error**

```
public void db_error(HttpServletRequest req,
                     HttpServletResponse resp,
                     String msg)
```

A database error has occurred

Overrides:

db_error in class ApplicationInterface

- **destroy**

```
protected void destroy()
```

Brush teeth before going to bed.

Overrides:

destroy in class ApplicationInterface

- **doc_access_error**

```
public void doc_access_error(HttpServletRequest req,
                             HttpServletResponse resp)
```

A document access error has occurred

Overrides:

doc_access_error in class ApplicationInterface

- **getCookie**

```
public java.lang.String getCookie(HttpServletRequest req)
```

Extract and return our cookie contents from the original HTTP request. The cookie used by this application holds the session id.

Parameters:

req - the original HTTP request.

Returns:

the session id associated with the request. If no session id is found returns null.

- **getCookieTag**

```
public java.lang.String getCookieTag()
```

- **getFullPath**

```
protected java.lang.String getFullPath(String document)
```

Given an HTML document name determine if a full path was specified. If not tack getProperty("ecardfile.html.base") on to the front of the document

Parameters:

document - the filename of the HTML document to be sent

Returns:

full URL of document

getOperation

```
public java.lang.String getOperation(HttpServletRequest req)
```

Determine the type of operation to be invoked by the request. Note that session creation and session destruction requests are indicated by the operation strings defined by RequestHandler.CREATE and RequestHandler.DESTROY respectively.

Parameters:

req - The HTTP request.

Returns:

A string describing the operation to carry out.

Overrides:

getOperation in class ApplicationInterface

●getPassword

```
public java.lang.String getPassword(HttpServletRequest req)
```

Get a string representing the user's password

Parameters:

req - The original HTTP request data.

Returns:

A string containing the password or null if the password tag isn't found in the HTTP request.

Overrides:

getPassword in class ApplicationInterface

●getServletImgBtnParameter

```
public java.lang.String
getServletImgBtnParameter(HttpServletRequest req) throws
ServletException, IOException
```

●getSessionId

```
public java.lang.String getSessionId(HttpServletRequest req)
throws ServletException, IOException
```

Extract and return the session id from the original HTTP request.

Parameters:

req - the original HTTP request.

Returns:

the session id associated with the request. If no session id is found returns null.

Throws: ServletException

Throws: IOException

Overrides:

getSessionId in class ApplicationInterface

●getUserId

```
public java.lang.String getUserId(HttpServletRequest req)
```

Get a string representing the user identification

Parameters:
req - The original HTTP request data.
Returns:
A string containing the user id or null if the user id tag isn't found in the HTTP request.
Overrides:
getUserId in class ApplicationInterface

- **hiddenField**

```
protected java.lang.String hiddenField(String name,
                                       String value)
```

Return a string of HTML specifying a hidden field.
Parameters:
name - NAME of the hidden field
value - VALUE of the hidden field
Returns:
HTML string as described above.

- **init**

```
public void init(ApplServlet servlet,
                 String managerName,
                 String rmiHost) throws ServletException,
IOException
```

Used to initialize the application specific class, which implements this interface. The servlet configuration is passed in so that the servlet environment is available to the application code.
Parameters:
applServlet - The Servlet instance which owns this ApplicationInterface instance.
Throws: ServletException
Overrides:
init in class ApplicationInterface

- **io_error**

```
public void io_error(HttpServletRequest req,
                     HttpServletResponse resp)
```

An io exception has occurred
Overrides:
io_error in class ApplicationInterface

- **isLoggedIn**

```
protected boolean isLoggedIn(String sessionId,
                             Session session) throws
SessionAccessException, IOException
```

Called to check if a user is logged in
Parameters:
sessionId - The Id of the current session

session - The current session

Returns:

True if the user is currently logged in

- **nfe_error**

```
public void nfe_error(HttpServletRequest req,
                      HttpServletResponse resp)
```

A NumberFormatException has occurred

Overrides:

nfe_error in class ApplicationInterface

- **nfe_error**

```
public void nfe_error(HttpServletRequest req,
                      HttpServletResponse resp,
                      String msg)
```

A NumberFormatException has occurred

Overrides:

nfe_error in class ApplicationInterface

- **nse_error**

```
public void nse_error(HttpServletRequest req,
                      HttpServletResponse resp)
```

A non-serializable exception has occurred

Overrides:

nse_error in class ApplicationInterface

- **operationRequiresLogin**

```
protected boolean operationRequiresLogin(String operation)
```

Called to check if a specified operation requires user login.

Parameters:

operation - The operation which was being attempted.

Returns:

True if the operation requires login, false otherwise

- **re_error**

```
public void re_error(HttpServletRequest req,
                     HttpServletResponse resp)
```

A Remote Exception has occurred

Overrides:

re_error in class ApplicationInterface

- **sae_error**

```
public void sae_error(HttpServletRequest req,
                      HttpServletResponse resp)
```

A Session Access Exception has occurred

Overrides:

sae_error in class ApplicationInterface

sendDocument

```
public void sendDocument(String document,
                         HttpServletResponse resp)
```

Send an HTML document to the user replacing the predefined character sequences with their associated values.

Parameters:

document - the file name of the HTML document to be sent. All documents are accessed relative to our ecardfile.html.base property

resp - provides methods to respond to the original HTTP request.

- **sendError**

```
public void sendError(String format,
                      String msg1,
                      String msg2,
                      HttpServletResponse resp)
```

- **sendError**

```
public void sendError(Hashtable tokens,
                      String format,
                      String msg1,
                      String msg2,
                      HttpServletResponse resp)
```

- **sendError**

```
public void sendError(Hashtable tokens,
                      String msg1,
                      String msg2,
                      HttpServletResponse resp)
```

- **sendError**

```
public void sendError(HttpServletRequest req,
                      String msg1,
                      String msg2,
                      HttpServletResponse resp)
```

General error handling routine.

Overrides:

sendError in class ApplicationInterface

- **sendLoginScreen**

```
protected void sendLoginScreen(HttpServletRequest req,
                               HttpServletResponse resp,
                               String format) throws IOException
```

- **sendLoginScreen**

```
protected void sendLoginScreen(HttpServletRequest req,
                               HttpServletResponse resp,
                               String format,
                               Hashtable initTokens) throws
IOException
```

sendMessage

```
public void sendMessage(String pdaPage,
                        String MessageFile,
                        Hashtable tokens,
                        String format,
                        HttpServletResponse resp)
```

- **sendMessage**

```
public void sendMessage(String MessageFile,
                        String format,
                        HttpServletResponse resp)
```

- **sendMessage**

```
public void sendMessage(String MessageFile,
                        Hashtable tokens,
                        String format,
                        HttpServletResponse resp)
```

- **sendParseDocument**

```
public void sendParseDocument(Hashtable tokenTable,
                              String document,
                              String format,
                              HttpServletResponse resp)
```

- **sendParseDocument**

```
public void sendParseDocument(Hashtable tokenTable,
                              String document,
                              HttpServletResponse resp)
```

Send an HTML document to the user replacing the predefined character sequences with their associated values.

Parameters:

tokenTable - the tokens and values to be replaced in the document
document - the file name of the HTML document to be sent. All documents are accessed relative to our ecardfile.html.base property session.
resp - provides methods to respond to the original HTTP request.

- **sendParseTextFile**

```
public void sendParseTextFile(String contentType,
                              String fileName,
                              Hashtable tokenTable,
                              String document,
                              String format,
                              HttpServletResponse resp)
```

- **sendParseTextFile**

```
public void sendParseTextFile(String contentType,
                              Hashtable tokenTable,
                              String document,
                              String format,
                              HttpServletResponse resp)
```

sendParseTextFile

```
public void sendParseTextFile(Hashtable tokenTable,
                              String document,
                              String format,
                              HttpServletResponse resp)
```

Send a text file to the user replacing the predefined character sequences with their associated values.

Parameters:

contentType - The content type of the text file
tokenTable - the tokens and values to be replaced in the document
document - the file name of the HTML document to be sent. All documents are accessed relative to our ecardfile.html.base property session.
format - the type of file to send, html or wml
resp - provides methods to respond to the original HTTP request.

- **sendSearchScreen**

```
protected void sendSearchScreen(HttpServletRequest req,
                                HttpServletResponse resp)
```

- **sendSearchScreen**

```
protected void sendSearchScreen(HttpServletRequest req,
                                HttpServletResponse resp,
                                String format) throws
IOException
```

- **sendSearchScreen**

```
protected void sendSearchScreen(HttpServletRequest req,
                                HttpServletResponse resp,
                                String format,
                                Hashtable initTokens) throws
IOException
```

- **sessionFailure**

```
public boolean sessionFailure(String sessionId,
                              HttpServletRequest req)
```

Implementation of ApplicationInterface:sessionFailure Notify the application that an attempt to access a session using sessionId failed. This could be due to a bogus sessionId or an expired session. We keep track of the failures per IP Address so that they can be checked in checkAccess()

Parameters:

req - The original HTTP request data.
sessionId - The session id string for the current session.

Returns:

If true a new session will be created

Overrides:

sessionFailure in class ApplicationInterface

unknownOperation

```
public void unknownOperation(HttpServletRequest req,
                             HttpServletResponse resp)
```

An unknown operation has been requested

Overrides:

unknownOperation in class ApplicationInterface

- **validateSession**

```
public boolean validateSession(String sessionId,
                               Session session,
                               HttpServletRequest req) throws
Exception
```

Validate the passed session This method can also be used to implement any security checks or other checks such as if undesirables are accessing the site - such as SPAMMERS

Parameters:

session - The session to validate.

req - The original HTTP request data.

Returns:

An indication if the session is valid

Overrides:

validateSession in class ApplicationInterface

- **verboseError**

```
protected java.lang.String verboseError(String msg)
```

Method which displays verbose error messages to user if debug is turned on

Class ecardfile.appl.EcardNotifier

```
Object
   |
   +---ecardfile.appl.EcardNotifier
```

public class **EcardNotifier**
extends Object
implements Runnable

Constructor Index

ecardfile.appl.EcardNotifier(CommonApplicationInterface, String, long, String)

Method Index

- interrupt()
- run()
- start()
- stop()

Constructors

EcardNotifier

```
public EcardNotifier(CommonApplicationInterface parent,
                     String templateCache,
                     long cardId,
                     String eCardId)
```

Methods

interrupt

```
public void interrupt()
```

run

```
public void run()
```

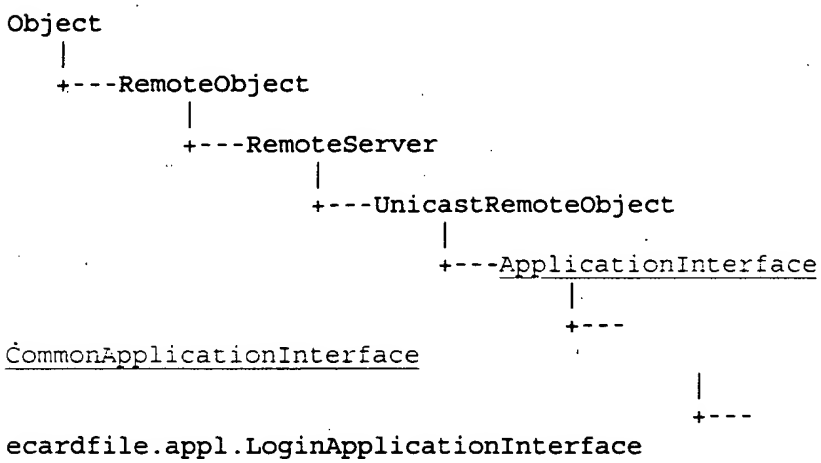
start

```
public void start()
```

stop

```
public void stop()
```

Class ecardfile.appl.LoginApplicationInterface



public class **LoginApplicationInterface**
extends CommonApplicationInterface
implements CommonConfig
The class implements the application behavior for the LoginServlet. Many of the methods are hooks called by RequestHandler instances invoked by LoginServlet (i.e. ApplServlet).

Version:

$Id: LoginApplicationInterface.java,v 1.23 1999/11/24 02:24:41 peter Exp $

See Also:

RequestHandler, LoginServlet, CommonConfig, SessionImpl

Constructor Index

- ecardfile.appl.LoginApplicationInterface()

Method Index

- accessDenied(String, HttpServletRequest, HttpServletResponse)
  Called when a received request does not contain a valid session id.
- authenticateUser(String, String, HttpServletRequest)
  Authenticate the user using a user id and password combination.
- executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler to execute an application defined operation.
- getCookieTag()
  Return the application specific cookie tag

- init(ApplServlet, String, String)
  Used to initialize the application specific class which implements this interface.
- notify(SessionNotification)
  Receives (asynchronous???) notifications from XXX This overrides the corresponding method in ApplicationInterface which in turn implements the method
- postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse)
  Post authentication functionality.
- postDestroy(HttpServletRequest, HttpServletResponse)
  Called by the RequestHandler immediately after the destruction of the session object.
- preDestroy(String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler prior to the destruction of the session object.

Constructors

LoginApplicationInterface

```
public LoginApplicationInterface() throws RemoteException
```

Methods

accessDenied

```
public void accessDenied(String operation,
                         HttpServletRequest req,
                         HttpServletResponse resp)
```

Called when a received request does not contain a valid session id. The application should return an error/warning document to the user. If the error occurred during a normal operation (i.e. other than Logging in or Logging out) it was more than likely due to the session having expired. In this case just bring up the login screen.

Parameters:

operation - The operation which was being attempted.
req - The original HTTP request.
resp - The HTTP response

Overrides:

accessDenied in class CommonApplicationInterface

authenticateUser

```
public multiserv.sessionmgr.GenericSession
authenticateUser(String userName,
String password,
HttpServletRequest req) throws RemoteException,
SessionAccessException, NotSerializableException,
ServletException, IOException
```

Authenticate the user using a user id and password combination. Authentication is done against the database. This method can also be used to implement any pre-session creation functionality.

Parameters:

userName - The user name string.

password - The user's password string.

req - The original HTTP request data.

Returns:

A session object containing any initial session data. The returned object must be a sub-class of GenericSession. This will be used to initialize the session object in the session manager. If the authentication fails, null is returned.

Throws: RemoteException

A problem occured while trying to create the session object in the SessionManager

Overrides:

authenticateUser in class ApplicationInterface

See Also:

UserAuth, SessionImpl

- **executeOperation**

```
public void executeOperation(String operation,
                             String sessionId,
                             Session session,
                             HttpServletRequest req,
                             HttpServletResponse resp)
```

Called by RequestHandler to execute an application defined operation. Valid operations are:

Arthroscopy

Performed on Peter's knee

Operations can be specification of the OP_TAG as a hidden field or as part of a Query String in a GET request. Alternatively, certain FORM buttons are defined to execute specific operations.

Parameters:

operation - Specifies the operation to perform. This should be one of the operation strings returned by getOperation.

session - An interface to the current session object.

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

executeOperation in class ApplicationInterface

- **getCookieTag**

```
public java.lang.String getCookieTag()
```

Return the application specific cookie tag

Returns:

name of the application specific cookie

Overrides:

getCookieTag in class CommonApplicationInterface

- **init**

```
public void init(ApplServlet servlet,
                 String managerName,
                 String rmiHost) throws ServletException,
IOException
```

Used to initialize the application specific class which implements this interface. The servlet configuration is passed in so that the servlet environment is available to the application code.

Parameters:

applServlet - The Servlet instance which owns this ApplicationInterface instance.

Throws: ServletException

Overrides:

init in class CommonApplicationInterface

- **notify**

```
public void notify(SessionNotification nofn)
```

Receives (asynchronous???) notifications from XXX This overrides the corresponding method in ApplicationInterface which in turn implements the method

Parameters:

nofn - An interface to the object carrying notification information.

Overrides:

notify in class ApplicationInterface

- **postAuthenticate**

```
public void postAuthenticate(String sessionId,
                             Session session,
                             HttpServletRequest req,
                             HttpServletResponse resp) throws
IOException, SessionAccessException, ServletException,
IOException
```

Post authentication functionality. If the authentication had failed a screen displaying an appropraite message is displayed.

Overrides:

postAuthenticate in class ApplicationInterface

- **postDestroy**

```
public void postDestroy(HttpServletRequest req,
                        HttpServletResponse resp)
```

Called by the RequestHandler immediately after the destruction of the session object. The session is finished. Display the Login screen.

Parameters:

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

postDestroy in class ApplicationInterface

- **preDestroy**

```
public void preDestroy(String sessionId,
                       Session session,
                       HttpServletRequest req,
                       HttpServletResponse resp)
```

Called by RequestHandler prior to the destruction of the session object.

Parameters:

session - An interface to the associated session object.

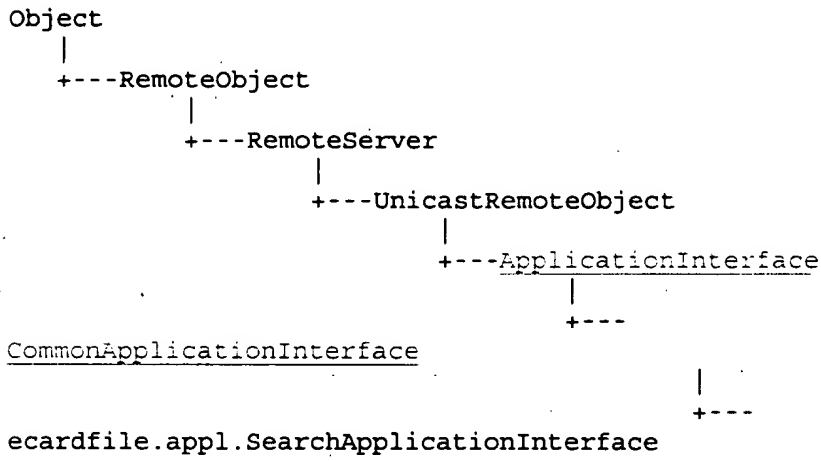
req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

preDestroy in class ApplicationInterface

Class ecardfile.appl.SearchApplicationInterface



public class **SearchApplicationInterface**
extends CommonApplicationInterface
implements CommonConfig
The class implements the application behavior for the SearchServlet. Many of the methods are hooks called by RequestHandler instances invoked by SearchServlet (i.e. ApplServlet).

Version:

$Id: SearchApplicationInterface.java,v 1.59 1999/12/03 01:10:09 peter Exp $

See Also:

RequestHandler, SearchServlet, CommonConfig, SessionImpl

Variable Index

- PRIVATE_ACCESS
- PUBLIC_ACCESS

Constructor Index

- ecardfile.appl.SearchApplicationInterface()

Method Index

- accessDenied(String, HttpServletRequest, HttpServletResponse)
  Called when a received request does not contain a valid session id.
- authenticateUser(String, String, HttpServletRequest)
  Authenticate the user using a user id and password combination.
- checkPrivacyAccess(short, Hashtable)

Checks all values in the card that is about to be displayed, if the privacy value for any value is higher than that granted by the card's owner the element is removed from the hashtable and the value is not displayed.

- destroy()
  Brush teeth before going to bed.
- executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler to execute an application defined operation.
- getCookieTag()
  Return the application specific cookie tag
- getPrivacyAccess(DatabaseConnection2, long, long)
  Get the privacy mask for the logged in user and the card being displayed
- getPrivacyAccess(DatabaseConnection2, long, long, Hashtable)
- init(ApplServlet, String, String)
  Used to initialize the application specific class, which implements this interface.
- notify(SessionNotification)
  Receives (asynchronous???) notifications from XXX This overrides the corresponding method in ApplicationInterface which in turn implements the method
- postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse)
  Post authentication functionality.
- postDestroy(HttpServletRequest, HttpServletResponse)
  Called by the RequestHandler immediately after the destruction of the session object.
- preDestroy(String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler prior to the destruction of the session object.

Variables

- PRIVATE_ACCESS

```
public static final short PRIVATE_ACCESS
```

- PUBLIC_ACCESS

```
public static final short PUBLIC_ACCESS
```

Constructors

- SearchApplicationInterface

```
public SearchApplicationInterface() throws RemoteException
```

Methods

- accessDenied

```
public void accessDenied(String operation,
                         HttpServletRequest req,
```

```
                                HttpServletResponse resp)
```

Called when a received request does not contain a valid session id. The application should return an error/warning document to the user. If the error occured during a normal operation (i.e. other than Logging in or Logging out) it was more than likely due to the session having expired. In this case just bring up the login screen.

Parameters:

operation - The operation which was being attempted.
req - The original HTTP request.
resp - The HTTP response

Overrides:

accessDenied in class CommonApplicationInterface

- **authenticateUser**

```
public multiserv.sessionmgr.GenericSession
authenticateUser(String userName,

String password,

HttpServletRequest req) throws RemoteException,
SessionAccessException, NotSerializableException,
ServletException, IOException
```

Authenticate the user using a user id and password combination. Authentication is done against the database. This method can also be used to implement any pre-session creation functionality.

Parameters:

userName - The user name string.
password - The user's password string.
req - The original HTTP request data.

Returns:

A session object containing any initial session data. The returned object must be a sub-class of GenericSession. This will be used to initialize the session object in the session manager. If the authentication fails, null is returned.

Throws: RemoteException

A problem occurred while trying to create the session object in the SessionManager

Overrides:

authenticateUser in class ApplicationInterface

See Also:

UserAuth, SessionImpl

- **checkPrivacyAccess**

```
public void checkPrivacyAccess(short privacyLevel,
                               Hashtable cardRow)
```

Checks all values in the card that is about to be displayed, if the privacy value for any value is higher than that granted by the card's owner the element is removed from the hashtable and the value is not displayed.

Parameters:

privacyLevel - Privacy value

cardRow - Hashtable from which the elements are removed

- **destroy**

```
protected void destroy()
```

Brush teeth before going to bed.

Overrides:

destroy in class CommonApplicationInterface

- **executeOperation**

```
public void executeOperation(String operation,
                             String sessionId,
                             Session session,
                             HttpServletRequest req,
                             HttpServletResponse resp)
```

Called by RequestHandler to execute an application defined operation.

Valid operations are:

Arthroscopy

Performed on Peter's knee

Operations can be specification of the OP_TAG as a hidden field or as part of a Query String in a GET request. Alternatively, certain FORM buttons are defined to execute specific operations.

Parameters:

operation - Specifies the operation to perform. This should be one of the operation strings returned by getOperation.

session - An interface to the current session object.

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

executeOperation in class ApplicationInterface

- **getCookieTag**

```
public java.lang.String getCookieTag()
```

Return the application specific cookie tag

Returns:

name of the application specific cookie

Overrides:

getCookieTag in class CommonApplicationInterface

- **getPrivacyAccess**

```
public short getPrivacyAccess(DatabaseConnection2 jdbc,
                              long loginId,
                              long cardId)
```

Get the privacy mask for the logged in user and the card being displayed

Parameters:

jdbc - An open database connection

loginId - user id of currently logged in user (0 if not logged in)

cardId - user id of the card details being checked

card - Card to add mask and privacy row id if required

- **getPrivacyAccess**

```
public void getPrivacyAccess(DatabaseConnection2 jdbc,
                             long loginId,
                             long cardId,
                             Hashtable card)
```

- **init**

```
public void init(ApplServlet servlet,
                 String managerName,
                 String rmiHost) throws ServletException,
IOException
```

Used to initialize the application specific class, which implements this interface. The servlet configuration is passed in so that the servlet environment is available to the application code.

Parameters:

applServlet - The Servlet instance which owns this ApplicationInterface instance.

Throws: ServletException

Overrides:

init in class CommonApplicationInterface

- **notify**

```
public void notify(SessionNotification nofn)
```

Receives (asynchronous???) notifications from XXX This overrides the corresponding method in ApplicationInterface which in turn implements the method

Parameters:

nofn - An interface to the object carrying notification information.

Overrides:

notify in class ApplicationInterface

- **postAuthenticate**

```
public void postAuthenticate(String sessionId,
                             Session session,
                             HttpServletRequest req,
                             HttpServletResponse resp) throws
IOException, SessionAccessException, ServletException
```

Post authentication functionality. If the authentication had failed a screen displaying an appropriate message is displayed.

Overrides:

postAuthenticate in class ApplicationInterface

postDestroy

```
public void postDestroy(HttpServletRequest req,
                        HttpServletResponse resp)
```

Called by the RequestHandler immediately after the destruction of the session object. The session is finished. Display the Login screen.

Parameters:

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

postDestroy in class ApplicationInterface

- **preDestroy**

```
public void preDestroy(String sessionId,
                       Session session,
                       HttpServletRequest req,
                       HttpServletResponse resp)
```

Called by RequestHandler prior to the destruction of the session object.

Parameters:

session - An interface to the associated session object.

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

Overrides:

preDestroy in class ApplicationInterface

package ecardfile.dbappl

Class Index

- Address
- Banner
- BannerCache
- DatabaseConnection2
- Email
- InactiveAddress
- InactiveDatabaseConnection
- InactiveEmail
- InactivePhone
- InactiveUser
- LockedIP
- Lookup
- LookupCache
- PersonalList
- Phone
- PrivateList
- User
- UserInfo
- UserObject
- WhereAmI

Class ecardfile.dbappl.Address

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.Address
```

public class **Address**
extends UserObject

Constructor Index

- ecardfile.dbappl.Address(Connection)

Constructors

- **Address**

```
public Address(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.Banner

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.Banner
```

public class **Banner**
extends JdbcObject

Constructor Index

ecardfile.dbappl.Banner(Connection)

Constructors

Banner

```
public Banner(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.BannerCache

```
Object
   |
   +---Cache
          |
          +---TableCache
                 |
                 +---ecardfile.dbappl.BannerCache
```

public class **BannerCache**
extends TableCache
Cache of the Banner table
See Also:
TableCache

Constructor Index

- ecardfile.dbappl.BannerCache()
  Shouldn't use this constructor
- ecardfile.dbappl.BannerCache(JdbcConnectionBroker2, long)

Method Index

- getJdbcObject(Connection)
  This method should be defined in the sub class.
- getRandomAd()
  Get a random Banner ad.
- populate()
  The database table is populated by using a JDBC object created by getJdbcObject.

Constructors

BannerCache

```
public BannerCache()
```

Shouldn't use this constructor

BannerCache

```
public BannerCache(JdbcConnectionBroker2 connMgr,
                   long secs) throws IOException
```

Parameters:
connMgr - - The JDBC Connection manager
secs - - The cache is repopulated every *secs* seconds

Methods

getJdbcObject

```
protected multiserv.dbmgr.JdbcObject getJdbcObject(Connection
connect) throws JdbcException
```

This method should be defined in the sub class.

Overrides:

getJdbcObject in class TableCache

●getRandomAd

```
public java.lang.String getRandomAd()
```

Get a random Banner ad. Basically it just returns an HTML String by using a random key into the banner table cache.

Returns:

String of HTML which can display an ad banner

See Also:

populate

●populate

```
public void populate()
```

The database table is populated by using a JDBC object created by getJdbcObject. We key a vector of keys locally. As records get added/deleted from the table the set of Ids will change. To retrieve random ads we randomly index into out local set of keys and use that key to pull an add from the cache.

Overrides:

populate in class TableCache

Class ecardfile.dbappl.DatabaseConnection2

```
Object
   |
   +---JdbcConnection
          |
          +---ecardfile.dbappl.DatabaseConnection2
```

public class **DatabaseConnection2**
extends JdbcConnection

Variable Index

- DEBUG

Constructor Index

- ecardfile.dbappl.DatabaseConnection2()
  Only to be used for JdbcConnectionBroker/JdbcConnectionFactory
- ecardfile.dbappl.DatabaseConnection2(String, String, String)

Method Index

- Address()
- Banner()
- Email()
- Initialize()
- LockedIP()
- Lookup()
- PersonalList()
- Phone()
- PrivateList()
- User()
- WhereAmI()
- close()
- getInstance(String, String, String)
- main(String[])

Variables

● DEBUG

```
public static boolean DEBUG
```

Constructors

● DatabaseConnection2

```
public DatabaseConnection2()
```
Only to be used for JdbcConnectionBroker/JdbcConnectionFactory

DatabaseConnection2

```
public DatabaseConnection2(String URL,
                           String username,
                           String password) throws JdbcException
```

Methods

Address

```
public ecardfile.dbappl.Address Address() throws JdbcException
```

Banner

```
public ecardfile.dbappl.Banner Banner() throws JdbcException
```

Email

```
public ecardfile.dbappl.Email Email() throws JdbcException
```

Initialize

```
public void Initialize() throws JdbcException
```

Overrides:

Initialize in class JdbcConnection

LockedIP

```
public ecardfile.dbappl.LockedIP LockedIP() throws JdbcException
```

Lookup

```
public ecardfile.dbappl.Lookup Lookup() throws JdbcException
```

PersonalList

```
public ecardfile.dbappl.PersonalList PersonalList() throws
JdbcException
```

Phone

```
public ecardfile.dbappl.Phone Phone() throws JdbcException
```

PrivateList

```
public ecardfile.dbappl.PrivateList PrivateList() throws
JdbcException
```

User

```
public ecardfile.dbappl.User User() throws JdbcException
```

WhereAmI

```
public ecardfile.dbappl.WhereAmI WhereAmI() throws JdbcException
```

close

```
public void close() throws SQLException
```

Overrides:

close in class JdbcConnection

getInstance

```
public multiserv.dbmgr.JdbcConnection getInstance(String URL,
                                                  String
username,
                                                  String
password) throws JdbcException
```

Overrides:

getInstance in class JdbcConnection

●main

```
public static void main(String[] args)
```

Class ecardfile.dbappl.Email

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.Email
```

public class **Email**
extends UserObject

Constructor Index

- ecardfile.dbappl.Email(Connection)

Constructors

- **Email**

```
public Email(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.InactiveAddress

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.InactiveAddress
```

public class **InactiveAddress**
extends UserObject

Constructor Index

- ecardfile.dbappl.InactiveAddress(Connection)

Constructors

- **InactiveAddress**

```
public InactiveAddress(Connection connect) throws JdbcException
```

Class
ecardfile.dbappl.InactiveDatabaseConnection

```
Object
   |
   +---ecardfile.dbappl.InactiveDatabaseConnection
```

public class **InactiveDatabaseConnection**
extends Object

Variable Index

- DEBUG

Constructor Index

- ecardfile.dbappl.InactiveDatabaseConnection(Connection)

Method Index

- InactiveAddress()
- InactiveEmail()
- InactivePhone()
- InactiveUser()

Variables

DEBUG

```
public static boolean DEBUG
```

Constructors

InactiveDatabaseConnection

```
public InactiveDatabaseConnection(Connection conn)
```

Methods

InactiveAddress

```
public ecardfile.dbappl.InactiveAddress InactiveAddress() throws
JdbcException
```

InactiveEmail

```
public ecardfile.dbappl.InactiveEmail InactiveEmail() throws
JdbcException
```

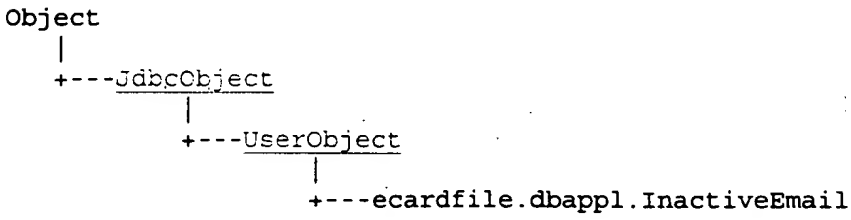
InactivePhone

```
public ecardfile.dbappl.InactivePhone InactivePhone() throws
JdbcException
```

InactiveUser

```
public ecardfile.dbappl.InactiveUser InactiveUser() throws
JdbcException
```

Class ecardfile.dbappl.InactiveEmail



public class **InactiveEmail**
extends UserObject

Constructor Index

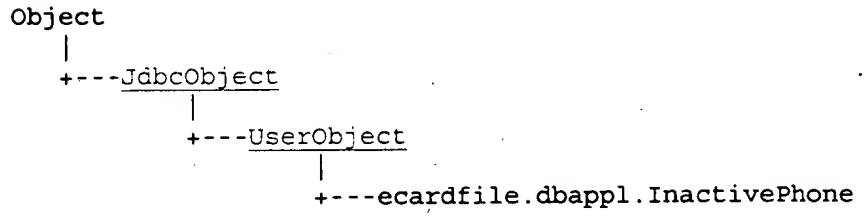
ecardfile.dbappl.InactiveEmail(Connection)

Constructors

InactiveEmail

```
public InactiveEmail(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.InactivePhone



public class **InactivePhone**
extends UserObject

Constructor Index

ecardfile.dbappl.InactivePhone(Connection)

Constructors

InactivePhone

```
public InactivePhone(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.InactiveUser

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.InactiveUser
```

public class **InactiveUser**
extends JdbcObject

Constructor Index

- ecardfile.dbappl.InactiveUser(Connection)

Method Index

- Delete(InactiveDatabaseConnection, long)
- Get(String)
- Get(String, String)
- Insert(InactiveDatabaseConnection, String[], Vector, Vector, Vector)
- Update(InactiveDatabaseConnection, String[], Vector, Vector, Vector)

Constructors

● **InactiveUser**

```
public InactiveUser(Connection connect) throws JdbcException
```

Methods

● **Delete**

```
public int Delete(InactiveDatabaseConnection database,
                  long userId) throws JdbcException
```

● **Get**

```
public java.util.Hashtable Get(String szECardId) throws
JdbcException
```

● **Get**

```
public java.util.Hashtable Get(String szECardId,
                               String szSessionId) throws
JdbcException
```

● **Insert**

```
public long Insert(InactiveDatabaseConnection database,
                   String[] userRow,
```

```
                    Vector addressRows,
                    Vector phoneRows,
                    Vector emailRows) throws JdbcException
```

Update

```
public long Update(InactiveDatabaseConnection database,
                    String[] userRow,
                    Vector addressRows,
                    Vector phoneRows,
                    Vector emailRows) throws JdbcException
```

Class ecardfile.dbappl.LockedIP

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.LockedIP
```

public class **LockedIP**
extends JdbcObject

Variable Index

- psAll

Constructor Index

- ecardfile.dbappl.LockedIP(Connection)

Method Index

- QueryAll()

Variables

● **psAll**

```
protected java.sql.PreparedStatement psAll
```

Constructors

● **LockedIP**

```
public LockedIP(Connection connect) throws JdbcException
```

Methods

● **QueryAll**

```
public java.util.Vector QueryAll() throws JdbcException
```

Overrides:
QueryAll **in class** JdbcObject

Class ecardfile.dbappl.Lookup

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.Lookup
```

public class **Lookup**
extends JdbcObject

Constructor Index

- ecardfile.dbappl.Lookup(Connection)

Constructors

- **Lookup**

```
public Lookup(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.LookupCache

```
Object
   |
   +---Cache
          |
          +---TableCache
                 |
                 +---ecardfile.dbappl.LookupCache
```

public class **LookupCache**
extends TableCache
Cache of the Lookup table
See Also:
TableCache

Variable Index

- ADDRESS
- EMAIL
- PHONE
- USERINFO

Constructor Index

- ecardfile.dbappl.LookupCache()
  Shouldn't use this constructor
- ecardfile.dbappl.LookupCache(JdbcConnectionBroker2, long)

Method Index

- addLookupTokens(Short, Hashtable)
  Put the Lookup tokens in the token Hashtable
- addVcardTokens(Short, Hashtable)
  Put the Vcard tokens in the token Hashtable
- getJdbcObject(Connection)
  This method should be defined in the sub class.
- populate()
  The database table is populated by using a JDBC object created by getJdbcObject.
- replaceLookupTokens(Short, String, int, Hashtable)
  Replace the Lookup tokens in the token Hashtable
- replaceVcardTokens(Short, String, int, Hashtable)
  Replace the Vcard tokens in the token Hashtable

Variables

ADDRESS

```
public static final java.lang.Short ADDRESS
```

- EMAIL

```
public static final java.lang.Short EMAIL
```

- PHONE

```
public static final java.lang.Short PHONE
```

- USERINFO

```
public static final java.lang.Short USERINFO
```

Constructors

- LookupCache

```
public LookupCache()
```

Shouldn't use this constructor

- LookupCache

```
public LookupCache(JdbcConnectionBroker2 connMgr,
                   long secs) throws IOException
```

Parameters:

connMgr - - The JDBC Connection manager

secs - - The cache is repopulated every *secs* seconds

Methods

- addLookupTokens

```
public void addLookupTokens(Short category,
                            Hashtable tokens)
```

Put the Lookup tokens in the token Hashtable

- addVcardTokens

```
public void addVcardTokens(Short category,
                           Hashtable tokens)
```

Put the Vcard tokens in the token Hashtable

- getJdbcObject

```
protected multiserv.dbmgr.JdbcObject getJdbcObject(Connection
connect) throws JdbcException
```

This method should be defined in the sub class.

Overrides:

getJdbcObject in class TableCache

- populate

```
public void populate()
```

The database table is populated by using a JDBC object created by getJdbcObject. We call populate from the super class then keep the data in an optimized form for local use. The form is as for a token table of the types.

Overrides:

populate in class TableCache

- **replaceLookupTokens**

```
public void replaceLookupTokens(Short category,
                                String prefix,
                                int allocatedRows,
                                Hashtable tokens)
```

Replace the Lookup tokens in the token Hashtable

- **replaceVcardTokens**

```
public void replaceVcardTokens(Short category,
                               String prefix,
                               int allocatedRows,
                               Hashtable tokens)
```

Replace the Vcard tokens in the token Hashtable

Class ecardfile.dbappl.PersonalList

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.PersonalList
```

public class **PersonalList**
extends UserObject

Constructor Index

ecardfile.dbappl.PersonalList(Connection)

Method Index

- Delete(DatabaseConnection2, long, long)
- DeleteByCardId(DatabaseConnection2, long)
  Note: Transaction should be setup around this method
- DeleteByUserId(DatabaseConnection2, long)
  Note: Transaction should be setup around this method
- Insert(DatabaseConnection2, long, long, long, short)
- IsCardThere(long, long)
- QueryContainsCard(long)
- QueryJoinByUserId(long)
- QueryJoinByUserIdName(long, char)

Constructors

PersonalList

```
public PersonalList(Connection connect) throws JdbcException
```

Methods

Delete

```
public int Delete(DatabaseConnection2 jdbc,
                  long personalListId,
                  long privateListId) throws JdbcException
```

DeleteByCardId

```
public int DeleteByCardId(DatabaseConnection2 jdbc,
                          long cardId) throws JdbcException
```

Note: Transaction should be setup around this method

- **DeleteByUserId**

```
public int DeleteByUserId(DatabaseConnection2 jdbc,
                          long userId) throws JdbcException
```

Note: Transaction should be setup around this method

- **Insert**

```
public long Insert(DatabaseConnection2 jdbc,
                   long listId,
                   long userId,
                   long cardId,
                   short mask) throws JdbcException
```

- **IsCardThere**

```
public boolean IsCardThere(long userId,
                           long cardId) throws JdbcException
```

- **QueryContainsCard**

```
public java.util.Vector QueryContainsCard(long cardId) throws
JdbcException
```

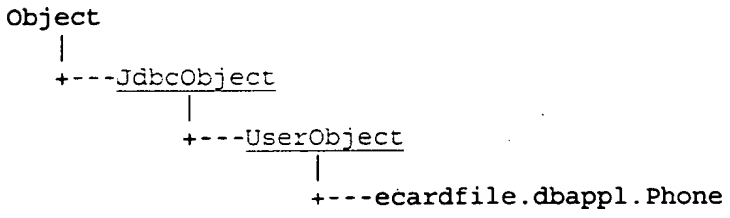
- **QueryJoinByUserId**

```
public java.util.Vector QueryJoinByUserId(long userId) throws
JdbcException
```

- **QueryJoinByUserIdName**

```
public java.util.Vector QueryJoinByUserIdName(long userId,
                                              char lastInitial)
throws JdbcException
```

Class ecardfile.dbappl.Phone



public class **Phone**
extends UserObject

Constructor Index

ecardfile.dbappl.Phone(Connection)

Constructors

Phone

```
public Phone(Connection connect) throws JdbcException
```

Class ecardfile.dbappl.PrivateList

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.PrivateList
```

public class **PrivateList**
extends UserObject

Constructor Index

- ecardfile.dbappl.PrivateList(Connection)

Method Index

- DeleteByCardId(long)
- Get(long, long)
- UpdateMask(long, short)

Constructors

- **PrivateList**

```
public PrivateList(Connection connect) throws JdbcException
```

Methods

- **DeleteByCardId**

```
public long DeleteByCardId(long cardId) throws JdbcException
```

- **Get**

```
public java.util.Hashtable Get(long loginId,
                               long cardId) throws JdbcException
```

- **UpdateMask**

```
public long UpdateMask(long privateListId,
                       short mask) throws JdbcException
```

Class ecardfile.dbappl.User

```
Object
   |
   +---JdbcObject
          |
          +---ecardfile.dbappl.User
```

public class **User**
extends JdbcObject

Constructor Index

- ecardfile.dbappl.User(Connection)

Method Index

- ConfirmUser(String, String)
- Delete(DatabaseConnection2, long)
- Get(String)
- GetForLogin(String, String)
- Insert(DatabaseConnection2, String[], Vector, Vector, Vector)
- QueryByFirstLastName(String, String)
- QueryByFirstLastNameSoundEx(String, String)
- Update(DatabaseConnection2, String[], Vector, Vector, Vector)

Constructors

- **User**

```
public User(Connection connect) throws JdbcException
```

Methods

- **ConfirmUser**

```
public int ConfirmUser(String eCardId,
                       String createId) throws JdbcException
```

- **Delete**

```
public int Delete(DatabaseConnection2 database,
                  long userId) throws JdbcException
```

- **Get**

```
public java.util.Hashtable Get(String szECardId) throws
JdbcException
```

- **GetForLogin**

```
public java.util.Hashtable GetForLogin(String szECardId,
                                       String szPassword) throws
JdbcException
```

Insert

```
public long Insert(DatabaseConnection2 database,
                   String[] userRow,
                   Vector addressRows,
                   Vector phoneRows,
                   Vector emailRows) throws JdbcException
```

●QueryByFirstLastName

```
public java.util.Vector QueryByFirstLastName(String szFirstName,
                                             String szLastName)
throws JdbcException
```

●QueryByFirstLastNameSoundEx

```
public java.util.Vector QueryByFirstLastNameSoundEx(String
szFirstName,
                                                    String
szLastName) throws JdbcException
```

●Update

```
public long Update(DatabaseConnection2 database,
                   String[] userRow,
                   Vector addressRows,
                   Vector phoneRows,
                   Vector emailRows) throws JdbcException
```

Class ecardfile.dbappl.UserInfo

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.UserInfo
```

public class **UserInfo**
extends UserObject

Constructor Index

- ecardfile.dbappl.UserInfo(Connection)

Method Index

- QueryByCategory(long, short)
- Update(long, Vector)

Constructors

UserInfo

```
public UserInfo(Connection connect) throws JdbcException
```

Methods

QueryByCategory

```
public java.util.Vector QueryByCategory(long cardId,
                                        short category) throws
JdbcException
```

Update

```
public long Update(long userId,
                   Vector rows) throws JdbcException
```

Class ecardfile.dbappl.UserObject

```
Object
   |
   +---JdbcObject
           |
           +---ecardfile.dbappl.UserObject
```

public class **UserObject**
extends JdbcObject

Constructor Index

- ecardfile.dbappl.UserObject(Connection, String, String[], int[], int[])

Method Index

- DeleteByUserId(long)
- Insert(long, String[])
- QueryByUserId(int)
- QueryByUserId(long)
- Update(long, String[])

Constructors

UserObject

```
public UserObject(Connection connect,
                  String table,
                  String[] columnNames,
                  int columnLength,
                  int columnTypes) throws JdbcException
```

Methods

DeleteByUserId

```
public long DeleteByUserId(long userId) throws JdbcException
```

Insert

```
public long Insert(long userId,
                   String[] row) throws JdbcException
```

QueryByUserId

```
public java.util.Vector QueryByUserId(int userId) throws
JdbcException
```

QueryByUserId

```
public java.util.Vector QueryByUserId(long userId) throws
JdbcException
```

Update

```
public long Update(long userId,
                   String[] row) throws JdbcException
```

Overrides:

Update in class JdbcObject

Class ecardfile.dbappl.WhereAmI

```
Object
   |
   +---JdbcObject
          |
          +---UserObject
                 |
                 +---ecardfile.dbappl.WhereAmI
```

public class **WhereAmI**
extends UserObject

Constructor Index

- ecardfile.dbappl.WhereAmI(Connection)

Method Index

- GetWithDate(long)
- QueryByExpiry(long, String)
- Update(long, String[])

Constructors

- **WhereAmI**

```
public WhereAmI(Connection connect) throws JdbcException
```

Methods

- **GetWithDate**

```
public java.util.Hashtable GetWithDate(long cardId) throws
JdbcException
```

- **QueryByExpiry**

```
public java.util.Vector QueryByExpiry(long cardId,
                                      String expDate) throws
JdbcException
```

- **Update**

```
public long Update(long userId,
                   String[] row) throws JdbcException
```

Overrides:
Update in class UserObject

package multiserv.applservlet

Class Index

- ApplServlet
- ApplicationInterface
- RequestHandler
- SessionTable

Class multiserv.applservlet.ApplServlet

```
Object
   |
   +---GenericServlet
           |
           +---HttpServlet
                   |
                   +---multiserv.applservlet.ApplServlet
```

public class **ApplServlet**
extends HttpServlet
The base class for all Servlets which use the Multi Server/Multi Servlet environment. Application servlets should extend this class.

For most cases, only the getApplicationInterface() needs to be overridden so that it returns the appropriate application specific implementation of ApplicationInterface.

Constructor Index

- multiserv.applservlet.ApplServlet()

Method Index

- decrCurrentCount()
- destroy()
- doGet(HttpServletRequest, HttpServletResponse)
  Overrides the doGet method provided by the HttpServlet superclass.
- doPost(HttpServletRequest, HttpServletResponse)
  Overrides the doPost method provided by the HttpServlet superclass.
- getApplicationInterface()
  Defines a method which returns an instance of ApplicationInterface.
- getProperty(String)
- getSessionMgr()
- incrCurrentCount()
- init(ServletConfig)
  Called when the servlet first gets loaded.
- log(String)
- setSessionMgrState(int)
- trace(String)

Constructors

- **ApplServlet**

```
public ApplServlet()
```

Methods

- **decrCurrentCount**

```
public synchronized void decrCurrentCount()
```

- **destroy**

```
public void destroy()
```

Overrides:
destroy in class GenericServlet

- **doGet**

```
public void doGet(HttpServletRequest req,
                  HttpServletResponse resp) throws
ServletException, IOException
```

Overrides the doGet method provided by the HttpServlet superclass. The service() method of HtpServlet handles the setup and dispatching to all doXXX() methods, which is why it usually should not be overridden.
Parameters:
req - The HTTP server request data.
resp - Provides methods to respond to the request.
Throws: ServletException
a problem occurred during the processing of the request.
Throws: IOException
an I/O problem was encountered.
Overrides:
doGet in class HttpServlet
See Also:
doPost

- **doPost**

```
public void doPost(HttpServletRequest req,
                   HttpServletResponse resp) throws
ServletException, IOException
```

Overrides the doPost method provided by the HttpServlet superclass. The service() method of HtpServlet handles the setup and dispatching to all doXXX() methods, which is why it usually should not be overridden.
Parameters:
req - The HTTP server request data.
resp - Provides methods to respond to the request.
Throws: ServletException
a problem occurred during the processing of the request.
Throws: IOException
an I/O problem was encountered.
Overrides:

doPost in class HttpServlet

See Also:

doGet

- **getApplicationInterface**

```
public multiserv.applservlet.ApplicationInterface
getApplicationInterface()
```

Defines a method which returns an instance of ApplicationInterface. This must be overridden in a subclass to provide an instance of a class which implements the methods defined in ApplicationInterface.

- **getProperty**

```
public java.lang.String getProperty(String propName)
```

- **getSessionMgr**

```
public synchronized multiserv.sessionmgr.SessionMgr
getSessionMgr() throws NotBoundException, RemoteException,
MalformedURLException
```

- **incrCurrentCount**

```
public synchronized void incrCurrentCount()
```

- **init**

```
public void init(ServletConfig config) throws ServletException
```

Called when the servlet first gets loaded. Do servlet initialization here. This specializes the init() method in GenericServlet.

Parameters:

config - servlet configuration information.

Throws: ServletException

a problem occurred during the initialization of the servlet.

Overrides:

init in class GenericServlet

- **log**

```
public void log(String msg)
```

Overrides:

log in class GenericServlet

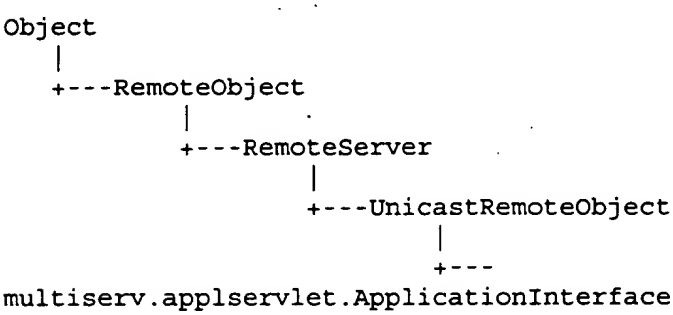
- **setSessionMgrState**

```
public synchronized void setSessionMgrState(int newState)
```

- **trace**

```
public void trace(String msg)
```

Class multiserv.applservlet.ApplicationInterface



public abstract class **ApplicationInterface**
extends UnicastRemoteObject
implements SessionObserver
Encapsulates the application dependent operations which are invoked from the RequestHandler class.

Constructor Index

- multiserv.applservlet.ApplicationInterface()

Method Index

- accessDenied(String, HttpServletRequest, HttpServletResponse)
  Called when a received request does not contain a valid session id.
- authenticateUser(String, String, HttpServletRequest)
  Authenticate the user using a user id and password combination.
- chainRequest(String, HttpServletRequest, HttpServletResponse)
  Method to forward a URL to another servlet.
- checkAccess(HttpServletRequest)
  Check the HTTP request data and decide if access should be allowed.
- db_error(HttpServletRequest, HttpServletResponse)
  A database error has occurred
- db_error(HttpServletRequest, HttpServletResponse, String)
  A database error has occurred
- destroy()
  Hook to get the ApplicationInterface to brush its teeth before going to bed.
- doc_access_error(HttpServletRequest, HttpServletResponse)
  A document access error has occurred
- executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler to execute an application defined operation.

- getOperation(HttpServletRequest)
  Determine the type of operation to be invoked by the request.
- getPassword(HttpServletRequest)
  Get a string representing the user's password
- getProperty(String)
- getServletParameter(HttpServletRequest, String)
  Convenience method to extract a parameter from a HttpServletRequest.
- getServletParameterValues(HttpServletRequest, String)
  Convenience method to a multi value parameter from a HttpServletRequest.
- getSessionId(HttpServletRequest)
  Extract and return the session id from the original HTTP request.
- getTimeZone()
- getUserId(HttpServletRequest)
  Get a string representing the user identification
- init(ApplServlet, String, String)
  Used to initialize the application specific class which implements this interface.
- io_error(HttpServletRequest, HttpServletResponse)
  An io exception has occurred
- log(String)
- nfe_error(HttpServletRequest, HttpServletResponse)
  A NumberFormatException has occurred
- nfe_error(HttpServletRequest, HttpServletResponse, String)
  A NumberFormatException has occurred
- notify(SessionNotification)
  Override this method to be receive notifications from the Session Manager.
- nse_error(HttpServletRequest, HttpServletResponse)
  A non serializable exception has occurred
- postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse)
  Post authentication functionality.
- postDestroy(HttpServletRequest, HttpServletResponse)
  Called by the RequestHandler immediately after the destruction of the session object.
- preDestroy(String, Session, HttpServletRequest, HttpServletResponse)
  Called by RequestHandler prior to the destruction of the session object.
- re_error(HttpServletRequest, HttpServletResponse)
  A Remote Exception has occurred
- sae_error(HttpServletRequest, HttpServletResponse)
  A Session Access Exception has occurred
- sendError(HttpServletRequest, String, String, HttpServletResponse)
  General error handling routine.
- sessionFailure(String, HttpServletRequest)

Notify the application that an attempt to access a session using sessionId failed.

- trace(String)
- unknownOperation(HttpServletRequest, HttpServletResponse)

  An unknown operation has been requested
- validateSession(String, Session, HttpServletRequest)

  Validate the passed session This method can also be used to implement any security checks or other checks such as if undesirables are accessing the site - such as SPAMMERS

Constructors

- **ApplicationInterface**

```
public ApplicationInterface() throws RemoteException
```

Methods

- **accessDenied**

```
public abstract void accessDenied(String operation,
                                  HttpServletRequest req,
                                  HttpServletResponse resp)
```

Called when a received request does not contain a valid session id. The application should return an error/warning document to the user.

Parameters:

operation - the operation being performed
req - the original HTTP request.
resp - provides methods to respond to the HTTP server.
format - the http format to reply in

- **authenticateUser**

```
public abstract multiserv.sessionmgr.GenericSession
authenticateUser(String userId,
String password,
HttpServletRequest req) throws Exception
```

Authenticate the user using a user id and password combination. This method can also be used to implement any pre-session creation functionality.

Parameters:

userId - The user id string.
password - The user's password string.
req - The original HTTP request data.

Returns:

A session object containing any initial session data. The returned object must be a sub-class of GenericSession. This will be used to initialize the session object in the session manager. If the authentication fails, null is returned.

chainRequest

```
public void chainRequest(String servletUrl,
                         HttpServletRequest req,
                         HttpServletResponse resp) throws
ServletException, IOException
```

Method to forward a URL to another servlet. This is kept as infrastructure so that it is easily changed to the most correct way of doing this.

Parameters:

servletUrl - The Servlet URL to which this request should be chained

req - The HTTP request from the current servlet which is being forwarded on.

resp - provides methods to respond to the HTTP server.

Throws: ServletException

Throws: IOException

See Also:

getServletParameter

- **checkAccess**

```
public abstract boolean checkAccess(HttpServletRequest req)
throws Exception
```

Check the HTTP request data and decide if access should be allowed. An application might want to check the IP address or other paramters in the request.

Parameters:

req - The original HTTP request data.

Returns:

An indication if access is to be granted

- **db_error**

```
public abstract void db_error(HttpServletRequest req,
                              HttpServletResponse resp)
```

A database error has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **db_error**

```
public abstract void db_error(HttpServletRequest req,
                              HttpServletResponse resp,
                              String msg)
```

A database error has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

destroy

```
protected abstract void destroy()
```

Hook to get the ApplicationInterface to brush its teeth before going to bed.

doc_access_error

```
public abstract void doc_access_error(HttpServletRequest req,
                                      HttpServletResponse resp)
```

A document access error has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **executeOperation**

```
public abstract void executeOperation(String operation,
                                      String sessionId,
                                      Session session,
                                      HttpServletRequest req,
                                      HttpServletResponse resp)
```

Called by RequestHandler to execute an application defined operation.

Parameters:

operation - Specifies the operation to perform. This should be one of the operation strings returned by getOperation.

session - An interface to the current session object.

sessionId - The session id string for the current session.

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

- **getOperation**

```
public abstract java.lang.String getOperation(HttpServletRequest
req)
```

Determine the type of operation to be invoked by the request. Note that session creation and session destruction requests are indicated by the operation strings defined by RequestHandler.CREATE and RequestHandler.DESTROY respectively.

Parameters:

req - The HTTP request.

Returns:

A string describing the operation to carry out.

- **getPassword**

```
public abstract java.lang.String getPassword(HttpServletRequest
req)
```

Get a string representing the user's password

Parameters:

req - The original HTTP request data.

Returns:

A string containing the password.

getProperty

```
public java.lang.String getProperty(String propName)
```

getServletParameter

```
public java.lang.String getServletParameter(HttpServletRequest
req,
                                            String parmName)
throws ServletException, IOException
```

Convenience method to extract a parameter from a HttpServletRequest. The method first checks to see if the parameter came from another servlet by trying to extract from the request using getAttribute(). If this fails it uses getParameter to extract the parameter. This order of checking means that a calling servlet can override parameters.

Parameters:

req - The HTTP request

parmName - The name of the parameter to extract

Returns:

The value of the parameter or null if not found

Throws: ServletException

Throws: IOException

See Also:

getServletParameter

- **getServletParameterValues**

```
public java.lang.String[]
getServletParameterValues(HttpServletRequest req,
                                                    String
parmName) throws ServletException, IOException
```

Convenience method to a multi value parameter from a HttpServletRequest. The method first checks to see if the parameter came from another servlet by trying to extract from the request using getAttribute(). If this fails it uses getParameterValues to extract the parameter. This order of checking means that a calling servlet can override parameters. The caller must be aware if the parameter is a multi value parameter.

Parameters:

req - The HTTP request

parmName - The name of the parameter to extract

Returns:

The value of the parameter or null if not found

Throws: ServletException

Throws: IOException

See Also:

getServletParameter

getSessionId

```
public abstract java.lang.String getSessionId(HttpServletRequest
req) throws Exception
```

Extract and return the session id from the original HTTP request.

Parameters:

req - the original HTTP request.

Returns:

the session id associated with the request.

- **getTimeZone**

```
protected java.util.TimeZone getTimeZone()
```

- **getUserId**

```
public abstract java.lang.String getUserId(HttpServletRequest
req)
```

Get a string representing the user identification

Parameters:

req - The original HTTP request data.

Returns:

A string containing the user id.

- **init**

```
public void init(ApplServlet applServlet,
                 String managerName,
                 String rmiHost) throws ServletException,
IOException
```

Used to initialize the application specific class which implements this interface. The servlet configuration is passed in so that the servlet environment is available to the application code.

Parameters:

applServlet - The Servlet instance which owns this ApplicationInterface instance.

managerName - The name of the session manager instance to map.

rmihost - The host on which the registry is running

- **io_error**

```
public abstract void io_error(HttpServletRequest req,
                              HttpServletResponse resp)
```

An io exception has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **log**

```
public void log(String message)
```

- **nfe_error**

```
public abstract void nfe_error(HttpServletRequest req,
                               HttpServletResponse resp)
```

A NumberFormatException has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **nfe_error**

```
public abstract void nfe_error(HttpServletRequest req,
                               HttpServletResponse resp,
                               String msg)
```

A NumberFormatException has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

msg - Additional info to be displayed

- **notify**

```
public void notify(SessionNotification nofn)
```

Override this method to be receive notifications from the Session Manager.

Parameters:

nofn - Interface which accesses notification information.

- **nse_error**

```
public abstract void nse_error(HttpServletRequest req,
                               HttpServletResponse resp)
```

A non serializable exception has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **postAuthenticate**

```
public abstract void postAuthenticate(String sessionId,
                                      Session initialSession,
                                      HttpServletRequest req,
                                      HttpServletResponse resp)
throws Exception
```

Post authentication functionality. This would probably include sending an HTML document back to the user.

Parameters:

sessionId - a string identifying the new client session.

initialSession - the new client Session object initialized with initial data.

req - the original HTTP request.

resp - provides methods to respond to the HTTP server.

- **postDestroy**

```
public abstract void postDestroy(HttpServletRequest req,
                                 HttpServletResponse resp)
```

Called by the RequestHandler immediately after the destruction of the session object.

Parameters:

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

- **preDestroy**

```
public abstract void preDestroy(String sessionId,
                                Session session,
                                HttpServletRequest req,
                                HttpServletResponse resp)
```

Called by RequestHandler prior to the destruction of the session object.

Parameters:

sessionId - The session id string for the current session.

session - An interface to the associated session object.

req - The data from the original HTTP request.

resp - Provides methods for responding to the request.

- **re_error**

```
public abstract void re_error(HttpServletRequest req,
                              HttpServletResponse resp)
```

A Remote Exception has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **sae_error**

```
public abstract void sae_error(HttpServletRequest req,
                               HttpServletResponse resp)
```

A Session Access Exception has occurred

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **sendError**

```
public abstract void sendError(HttpServletRequest req,
                               String msg1,
                               String msg2,
                               HttpServletResponse resp)
```

General error handling routine. This will force the class user to implement error messages

Parameters:

req - The HTTP request

msg1 - First message to be displayed

msg2 - Second message to be displayed

resp - The HTTP response

sessionFailure

```
public abstract boolean sessionFailure(String sessionId,
                                       HttpServletRequest req)
throws Exception
```

Notify the application that an attempt to access a sesssion using sessionId failed. This could be due to a Bogus sessionId or an expired session. The application will decide if a new session is to be created or if accessDenied() should be called. An application might want to log the failure or implement some sort of safeguard such as blocking the IP Address after a number of failures.

Parameters:

req - The original HTTP request data.

sessionId - The session id string for the current session.

Returns:

If true a new session will be created

- **trace**

```
public void trace(String message)
```

- **unknownOperation**

```
public abstract void unknownOperation(HttpServletRequest req,
                                      HttpServletResponse resp)
```

An unknown operation has been requested

Parameters:

format - the predefined format to reply in

resp - The HTTP response

- **validateSession**

```
public abstract boolean validateSession(String sessionId,
                                        Session session,
                                        HttpServletRequest req)
throws Exception
```

Validate the passed session This method can also be used to implement any security checks or other checks such as if undesirables are accessing the site - such as SPAMMERS

Parameters:

session - The session to validate.

req - The original HTTP request data.

Returns:

An indication if the session is valid

Class multiserv.applservlet.RequestHandler

```
Object
   |
   +---multiserv.applservlet.RequestHandler
```

public class **RequestHandler**
extends Object
A RequestHandler instance is created for each HTTP request received by the Application Servlet.

Variable Index

- CREATE
  The predefined operation types which can be executed by RequestHandler.
- DESTROY
- applInterface
  An instance of an application specific class which implements the ApplicationInterface
- managerName
  The name of the remote Session Manager being used
- operation
  Shows which operation this thread is executing
- request
  Holds the data from the original HTTP request
- response
  Provides methods for responding to the request
- rmiHost
  The hostname of the RMI registry
- servlet
  The parent Servlet instance which created this RequestHandler

Constructor Index

- multiserv.applservlet.RequestHandler(String, String, ApplServlet, ApplicationInterface, HttpServletRequest, HttpServletResponse)
  The constructor.

Method Index

- handle()
  The method which implements the request handling functionality.
- sessionMgr()

- sessionObjName(String)
  Constructs the session object name given the session id.

Variables

- CREATE

```
public static final java.lang.String CREATE
```

  The predefined operation types which can be executed by RequestHandler. All other operation types are application dependent and must be implemented using ApplicationInterface (or extension thereof)

- DESTROY

```
public static final java.lang.String DESTROY
```

- applInterface

```
protected multiserv.applservlet.ApplicationInterface
applInterface
```

  An instance of an application specific class which implements the ApplicationInterface

- managerName

```
protected java.lang.String managerName
```

  The name of the remote Session Manager being used

- operation

```
protected java.lang.String operation
```

  Shows which operation this thread is executing

- request

```
protected javax.servlet.http.HttpServletRequest request
```

  Holds the data from the original HTTP request

- response

```
protected javax.servlet.http.HttpServletResponse response
```

  Provides methods for responding to the request

- rmiHost

```
protected java.lang.String rmiHost
```

  The hostname of the RMI registry

- servlet

```
protected multiserv.applservlet.ApplServlet servlet
```

  The parent Servlet instance which created this RequestHandler

Constructors

- RequestHandler

```
public RequestHandler(String mgrName,
```

```
                        String rmihost,
                        ApplServlet applServlet,
                        ApplicationInterface appl,
                        HttpServletRequest req,
                        HttpServletResponse resp) throws
RemoteException, MalformedURLException, NotBoundException
```

The constructor. Initializes the instances data members.

Parameters:

mgrName - The name of the session manager instance to map.
rmihost - The host on which the registry is running
applServlet - The parent ApplServlet instance.
appl - The application specifics
req - The request data from the HTTP server.
resp - Provides methods for responding to the request.

Throws: RemoteException
if registry could not be contacted.

Throws: NotBoundException
if SessionMgr is not currently bound.

Methods

- **handle**

```
public void handle()
```

The method which implements the request handling functionality. This method processes requests and invokes the appropriate application level methods in ApplicationInterface.

- **sessionMgr**

```
protected multiserv.sessionmgr.SessionMgr sessionMgr() throws
NotBoundException, RemoteException, MalformedURLException
```

- **sessionObjName**

```
protected java.lang.String sessionObjName(String sid)
```

Constructs the session object name given the session id.

Parameters:

sid - the session id string for the object

Returns:

a string containing the session object name or the empty string if sid is null.

Class multiserv.applservlet.SessionTable

```
Object
   |
   +---Dictionary
           |
           +---Hashtable
                   |
                   +---multiserv.applservlet.SessionTable
```

public class **SessionTable**
extends Hashtable
A map containing the remote session object interfaces. Each interface is keyed using it's associated session id string.

Constructor Index

- multiserv.applservlet.SessionTable()

Method Index

- getSession(String)
  Retrieve a Session interface using the associated session id string.
- putSession(Session, String)
  Insert a Session interface into the map using the associated session id as the key.

Constructors

- SessionTable

```
public SessionTable()
```

Methods

- **getSession**

```
public multiserv.sessionmgr.Session getSession(String sessionId)
```

Retrieve a Session interface using the associated session id string.

Parameters:

sessionId - The session id string.

Returns:

The Session interface keyed to the given session id or null if the key is invalid.

- **putSession**

```
public void putSession(Session sess,
```

String sessionId)

Insert a Session interface into the map using the associated session id as the key.

Parameters:

sess - The remote session object interface.

sessionId - The session id string.

package multiserv.dbmgr

Class Index

- JdbcConnection
- JdbcConnectionBroker2
- JdbcConnectionFactory
- JdbcObject
- JdbcVendor
- TableCache
- Timing

Exception Index

- JdbcException

Class multiserv.dbmgr.JdbcConnection

```
Object
   |
   +---multiserv.dbmgr.JdbcConnection
```

public abstract class **JdbcConnection**
extends Object

Variable Index

- debug

Constructor Index

- multiserv.dbmgr.JdbcConnection()
  Default constructor only used for JdbcConnectionFactory
- multiserv.dbmgr.JdbcConnection(String, String, String)

Method Index

- Connect(String, String, String)
- Initialize()
- clearWarnings()
- close()
- createStatement()
- getConnection()
- getInstance(String, String, String)
- getWarnings()
- isClosed()
- toString()

Variables

debug

```
public static boolean debug
```

Constructors

JdbcConnection

```
public JdbcConnection()
```

Default constructor only used for JdbcConnectionFactory

JdbcConnection

```
protected JdbcConnection(String URL,
                         String username,
                         String password) throws JdbcException
```

Methods

- **Connect**

```
public boolean Connect(String URL,
                       String username,
                       String password)
```

- **Initialize**

```
protected abstract void Initialize() throws JdbcException
```

- **clearWarnings**

```
public void clearWarnings() throws SQLException
```

- **close**

```
public void close() throws SQLException
```

- **createStatement**

```
public java.sql.Statement createStatement() throws SQLException
```

- **getConnection**

```
public java.sql.Connection getConnection()
```

- **getInstance**

```
public abstract multiserv.dbmgr.JdbcConnection
getInstance(String URL,String username,String password) throws
JdbcException
```

- **getWarnings**

```
public java.sql.SQLWarning getWarnings() throws SQLException
```

- **isClosed**

```
public boolean isClosed() throws SQLException
```

- **toString**

```
public java.lang.String toString()
```

Overrides:
toString in class Object

Class multiserv.dbmgr.JdbcConnectionBroker2

```
Object
   |
   +----multiserv.dbmgr.JdbcConnectionBroker2
```

public class **JdbcConnectionBroker2**
extends Object
implements Runnable
JdbcConnectionBroker2 A servlet-based broker for database connections. Creates and manages a pool of database connections.

Version:

1.0.7 9/19/98

Author:

Marc A. Mnich

Constructor Index

-multiserv.dbmgr.JdbcConnectionBroker2(String, String, String, String, JdbcConnection, int, int, String, double)
Creates a new Connection Broker
dbDriver: JDBC driver.

Method Index

- destroy()
  Shuts down the housekeeping thread and closes all connections in the pool.
- freeConnection(JdbcConnection)
  Frees a connection.
- getAge(JdbcConnection)
  Returns the age of a connection -- the time since it was handed out to an application.
- getConnection()
  This class hands out the connections in round-robin order.
- idOfConnection(JdbcConnection)
  Returns the local JDBC ID for a connection.
- interrupt()
  A hook for future expansion.
- release()
  Release was method used in previous Connection Pool manager
- run()
  Housekeeping thread.

Constructors

- JdbcConnectionBroker2

```
public JdbcConnectionBroker2(String dbDriver,
                             String dbServer,
                             String dbLogin,
                             String dbPassword,
                             JdbcConnection connection,
                             int minConns,
                             int maxConns,
                             String logFileString,
                             double maxConnTime) throws
IOException
```

Creates a new Connection Broker
dbDriver: JDBC driver. e.g. 'oracle.jdbc.driver.OracleDriver'
dbServer: JDBC connect string. e.g.
'jdbc:oracle:thin:@203.92.21.109:1526:orcl'
dbLogin: Database login name. e.g. 'Scott'
dbPassword: Database password. e.g. 'Tiger'
minConns: Minimum number of connections to start with.
maxConns: Maximum number of connections in dynamic pool.
logFileString: Absolute path name for log file. e.g. 'c:\\temp\\mylog.log'
maxConnTime: Time in days between connection resets. (Reset does a basic cleanup)

Methods

- destroy

```
public void destroy()
```

Shuts down the housekeeping thread and closes all connections in the pool. Call this method from the destroy() method of the servlet.

- freeConnection

```
public java.lang.String freeConnection(JdbcConnection conn)
```

Frees a connection. Replaces connection back into the main pool for resuse.

- getAge

```
public long getAge(JdbcConnection conn)
```

Returns the age of a connection -- the time since it was handed out to an application.

- getConnection

```
public multiserv.dbmgr.JdbcConnection getConnection()
```

This class hands out the connections in round-robin order. This prevents a faulty connection from locking up an application entirely. A browser 'refresh' will get the next connection while the faulty connection is cleaned

up by the housekeeping thread. If the min number of threads are ever exhausted, new threads are added up the the max thread count. Finally, if all threads are in use, this method waits 2 seconds and tries again, up to ten times. After that, it returns a null.

- **idOfConnection**

```
public int idOfConnection(JdbcConnection conn)
```

Returns the local JDBC ID for a connection.

- **interrupt**

```
public void interrupt()
```

A hook for future expansion. Currently it is used to interrupt the housekeeping thread.

- **release**

```
public void release()
```

Release was method used in previous Connection Pool manager

- **run**

```
public void run()
```

Housekeeping thread. Runs in the background with low CPU overhead. Connections are checked for warnings and closure and are periodically restarted. This thread is a catchall for corrupted connections and prevents the buildup of open cursors. (Open cursors result when the application fails to close a Statement). This method acts as fault tolerance for bad connection/statement programming.

Class multiserv.dbmgr.JdbcConnectionFactory

```
Object
   |
   +---multiserv.dbmgr.JdbcConnectionFactory
```

public class JdbcConnectionFactory
extends Object

Constructor Index

•multiserv.dbmgr.JdbcConnectionFactory(JdbcConnection, String, String, String)

Method Index

•getConnection()

Constructors

•JdbcConnectionFactory

```
public JdbcConnectionFactory(JdbcConnection connection,
                             String URL,
                             String username,
                             String password)
```

Methods

●getConnection

```
public multiserv.dbmgr.JdbcConnection getConnection() throws
JdbcException
```

Class multiserv.dbmgr.JdbcObject

```
Object
   |
   +---multiserv.dbmgr.JdbcObject
```

public class JdbcObject
extends Object
The base for a JdbcObject. When specifying the column names the tableId, if used, must be specified as the first column as *table*Id. E.g. For a table called Users and there is an id column it must be specified as the first column as UsersId.

Variable Index

Constructor Index

Method Index

Variables

COLUMNS

protected static final int COLUMNS
COLUMNS_INSERT

protected static final int COLUMNS_INSERT
DEBUG

public static boolean DEBUG
INSERTS

protected static final int INSERTS
UPDATES

protected static final int UPDATES
VALUES

protected static final int VALUES
connect

protected java.sql.Connection connect
ps

protected java.sql.PreparedStatement ps
psAll

protected java.sql.PreparedStatement psAll
psById

protected java.sql.PreparedStatement psById
psDelete

protected java.sql.PreparedStatement psDelete
psInsert

protected java.sql.PreparedStatement psInsert
psMaxId

protected java.sql.PreparedStatement psMaxId
psUpdate

protected java.sql.PreparedStatement psUpdate
psUpdateCheck

protected java.sql.PreparedStatement psUpdateCheck

Constructors

JdbcObject

```
public JdbcObject(Connection connect,
                  String table,
                  String[] columnNames,
                  int columnLength,
                  int columnTypes) throws JdbcException
```

Methods

- Delete

```
public long Delete(long id) throws JdbcException
```

- Execute

```
public long Execute() throws JdbcException
```

- Get

```
public java.util.Hashtable Get() throws JdbcException
```

- Get

```
public java.util.Hashtable Get(long lId) throws JdbcException
```

- Insert

```
public long Insert(String[] row) throws JdbcException
```

- Query

```
public java.util.Vector Query() throws JdbcException
```

- QueryAll

```
public java.util.Vector QueryAll() throws JdbcException
```

- TableName

```
public java.lang.String TableName()
```

- Update

```
public long Update(long id,
                   String[] row) throws JdbcException
```

- UpdateCheck

```
public long UpdateCheck(long id,
                        long checkId,
                        String[] row) throws JdbcException
```

- UpdateRow

```
public long UpdateRow(String[] row) throws JdbcException
```

- getColumnNames

```
public java.lang.String[] getColumnNames()
```

- getColumnString

```
public java.lang.String getColumnString(int iType)
```

Generate a string containing the column names given a particular operation type. Valid operation types are:
COLUMNS
All of the columns
COLUMNS_INSERT
Columns for an insert. Note that this will not include the tableId column if JdbcVendor.SUPPORT_ID is turned on
UPDATES
Columns for an update. Note that this will not include the tableId column if JdbcVendor.SUPPORT_ID is turned on
VALUES
Columns for a VALUES clause of an INSERT statement. Note that this will not include the tableId column if JdbcVendor.SUPPORT_ID is turned on

Parameters:
iType - Indicate the operation type for which the column string is being generated

getColumnTypes

```
public int[] getColumnTypes()
```

getRow

```
public java.util.Hashtable getRow(ResultSet results,
                                  ResultSetMetaData meta,
                                  int cols)
```

Class multiserv.dbmgr.JdbcVendor

```
Object
   |
   +---multiserv.dbmgr.JdbcVendor
```

public class **JdbcVendor**
extends Object
All of the vendor specific stuff goes in here. Things like supporting an automatically incrementing row id column. Edit this file and recompile ... yuk!

Variable Index

Constructor Index

Method Index

Variables

DUP_INDEX

```
public static int DUP_INDEX
```

DUP_VALUE

```
public static int DUP_VALUE
```

INFORMIX When inserting a row and a unique column constraint is violated this is the error code returned in getErrorCode() from the SQLException

SUPPORT_ID

```
public static boolean SUPPORT_ID
```

Some call it SERIAL (eg Informix), some call it IDENTITY others just don't have it. Its a column which automatically creates a serial id for a row. Postgres doesn't have it but come to thing of it there is another sort of Id for a row - the details escape me at the moment but that could be an option in the Postgres rather than the getNextId() method

Constructors

JdbcVendor

public JdbcVendor()

Methods

- **duplicateIndex**

public static boolean duplicateIndex(SQLException e)

- **getId**

public static long getId(PreparedStatement pStmt) throws SQLException

- **knownError**

public static boolean knownError(SQLException e)

- **setLockModeWait**

public static boolean setLockModeWait(Connection connect, int seconds) throws JdbcException

---

Class multiserv.dbmgr.TableCache

```
Object
   |
   +---Cache
          |
          +---multiserv.dbmgr.TableCache
```

public abstract class **TableCache**
extends Cache
This class provides a base class for caching a JdbcObject. Subclasses simply need to override the getJdbcObject(Connection) method
Version:
$Id: TableCache.java,v 1.4 1999/12/03 19:42:22 peter Exp $
See Also:
getJdbcObject, Cache

Constructor Index

- multiserv.dbmgr.TableCache()
- multiserv.dbmgr.TableCache(JdbcConnectionBroker2)
- multiserv.dbmgr.TableCache(JdbcConnectionBroker2, long)

Method Index

- getJdbcObject(Connection)
  This method should be defined in the sub class.
- getRow(long)
  Retrieve a row from the cached table by specifying the row id
- populate()
  The database table is populated by using a JDBC object created by getJdbcObject.
- reinitialize()
- repopulate()
  Simply calls populate.
- setConnManager(JdbcConnectionBroker2)

Constructors

TableCache

```
public TableCache()
```

TableCache

```
public TableCache(JdbcConnectionBroker2 connMgr) throws
IOException
```

Parameters:
connMgr - - The JDBC Connection broker

TableCache

```
public TableCache(JdbcConnectionBroker2 connMgr,
                  long secs) throws IOException
```

Parameters:
connMgr - - The JDBC Connection broker

secs - - The cache is repopulated every secs seconds

Methods

- **getJdbcObject**

```
protected abstract multiserv.dbmgr.JdbcObject
getJdbcObject(Connection connect) throws JdbcException
```

This method should be defined in the sub class. It should just create an instance of a subclasses JdbcObject, passing it connect.
Parameters:
connect - Database connection.

- **getRow**

```
public java.util.Hashtable getRow(long id) throws CacheException
```

Retrieve a row from the cached table by specifying the row id
Parameters:
id - The id of the row to be retrieved.
Returns:
Hashtable as returned by a JdbcObject single row query.
Throws: CacheException
- row not found in Cache

- **populate**

```
public void populate()
```

The database table is populated by using a JDBC object created by getJdbcObject.
Overrides:
populate in class Cache

- **reinitialize**

```
public void reinitialize()
```

Overrides:
reinitialize in class Cache

- **repopulate**

```
public void repopulate()
```

Simply calls populate. Obviously want to have a long cache check time. Could put in some sort of check to see when the table was last updated but this would be some sort of Vendor specific hook.
Overrides:
repopulate in class Cache

- **setConnManager**

```
public void setConnManager(JdbcConnectionBroker2 connMgr)
```

Parameters:
connMgr - - The JDBC Connection broker

All Packages Class Hierarchy This Package Previous Next
Index

Class multiserv.dbmgr.Timing

```
Object
   |
   +---multiserv.dbmgr.Timing
```

public class **Timing**
extends Object

Constructor Index

- multiserv.dbmgr.Timing()

Method Index

- LogTiming(String)

Constructors

- **Timing**

```
public Timing()
```

Methods

- **LogTiming**

```
public void LogTiming(String szMessage)
```

Class multiserv.dbmgr.JdbcException

```
Object
   |
   +---Throwable
           |
           +---Exception
                   |
                   +---SQLException
                           |
                           +---multiserv.dbmgr.JdbcException
```

public class JdbcException
extends SQLException

Constructor Index

- multiserv.dbmgr.JdbcException()
- multiserv.dbmgr.JdbcException(String)
- multiserv.dbmgr.JdbcException(String, String, int)

Method Index

- isDuplicateValue()

Constructors

JdbcException

```
public JdbcException()
```

JdbcException

```
public JdbcException(String s)
```

JdbcException

```
public JdbcException(String message,
                     String state,
                     int error)
```

Methods

isDuplicateValue

```
public boolean isDuplicateValue()
```

package multiserv.sessionmgr

Interface Index

- Session
- SessionMgr
- SessionNotification
- SessionObserver
- SessionTags

Class Index

- GenericSession
- NotificationData
- ObserverMap
- SessionExpirer
- SessionIdFactory
- SessionImpl
- SessionMap
- SessionMgrConnection
- SessionMgrImpl
- SessionMgrShutdown
- SocketHandler

Exception Index

- SessionAccessException

Interface multiserv.sessionmgr.Session

public abstract interface Session
extends Remote
The Session remote interface is used to access the SessionImpl objects within the session manager.

Method Index

Methods

expire

```
public abstract void expire() throws RemoteException
```

Mark this object as expired.

getDouble

```
public abstract double getDouble(String fieldName) throws
SessionAccessException, RemoteException
```

Get the value of a double precision floating point field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

Field doesn't exist or is private.

Throws: RemoteException

Communications error.

getInt

```
public abstract int getInt(String fieldName) throws
SessionAccessException, RemoteException
```

Get the value of an integer type field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

Field doesn't exist or is private.

Throws: RemoteException

Communications error.

getLastAccessed

```
public abstract long getLastAccessed() throws RemoteException
```

Get the time at which this instance was last accessed.

Returns:

the time in milliseconds since EPOCH when this object was last accessed.

getLong

```
public abstract long getLong(String fieldName) throws
SessionAccessException, RemoteException
```

Get the value of a long integer field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

Field doesn't exist or is private.
Throws: RemoteException
Communications error.

- **getObject**

```
public abstract java.lang.Object getObject(String fieldName)
throws SessionAccessException, RemoteException,
NotSerializableException
```

Get the value of a field within the associated SessionImpl instance. The field type must be an extension of Object and must also implement the Serializable interface.
Parameters:
fieldName - A string specifying the name of the field to set.
Returns:
The value of the field.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.
Throws: NotSerializableException
value doesn't implement Serializable

- **hasExpired**

```
public abstract boolean hasExpired() throws RemoteException
```

Determine if this object has expired
Returns:
true if the object has expired, false otherwise.

- **setDouble**

```
public abstract void setDouble(String fieldName,
                               double value) throws
SessionAccessException, RemoteException
```

Set the value of a double precision floating point field within the associated SessionImpl instance.
Parameters:
fieldName - A string specifying the name of the field to set.
value - The value to set the field to.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.

- **setInt**

```
public abstract void setInt(String fieldName,
                            int value) throws
SessionAccessException, RemoteException
```

Set the value of an integer type field within the associated SessionImpl instance.

Parameters:
fieldName - A string specifying the name of the field to set.
value - The value to set the field to.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.

- **setLong**

```
public abstract void setLong(String fieldName,
                             long value) throws
SessionAccessException, RemoteException
```

Set the value of a long integer field within the associated SessionImpl instance.

Parameters:
fieldName - A string specifying the name of the field to set.
value - The value to set the field to.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.

- **setObject**

```
public abstract void setObject(String fieldName,
                               Object value) throws
SessionAccessException, RemoteException,
NotSerializableException
```

Set the value of a field within the associated SessionImpl instance. The field type must be an extension of Object and must also implement the Serializable interface.

Parameters:
fieldName - A string specifying the name of the field to set.
value - The value to set the field to.
Throws: SessionAccessException
Field doesn't exist or is private.
Throws: RemoteException
Communications error.
Throws: NotSerializableException
value doesn't implement Serializable

- **touch**

```
public abstract void touch() throws RemoteException
```

Mark the time when this object was last accessed as now.

Interface multiserv.sessionmgr.SessionMgr

public abstract interface SessionMgr
extends Remote
The SessionMgr remote interface is used to control the session manager.

Method Index

- closeSession(String)
  Remove an existing SessionImpl object from the session manager.
- initSession(Session)
  Create a new SessionImpl object within the session manager.
- register(String)
  A remote entity registers interest in events taking place within the session manager by calling this method.
- sessionIdList()
  Return an array containing all the current Session object identifiers.
- shutdown(String)
  A remote entity informs the Session Manager that it is prepared to have the Manager shutdown operations.
- unregister(String)
  A remote entity calls this method to stop further notifications being sent by the session manager.

Methods

closeSession

```
public abstract void closeSession(String sessionId) throws
RemoteException, MalformedURLException, NotBoundException
```

Remove an existing SessionImpl object from the session manager.

Parameters:
sessionId - A string which identifies the session to be removed.

Throws: RemoteException
if some communication failure occurs.

Throws: NotBoundException
the session which is being closed does not have it's interface bound to the RMI registry.

initSession

```
public abstract java.lang.String initSession(Session session)
throws RemoteException, MalformedURLException,
SessionAccessException, NotSerializableException
```

Create a new SessionImpl object within the session manager.

Parameters:

session - Contains the initial session data.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

Throws: RemoteException

if some communication failure occurs.

- **register**

```
public abstract void register(String id) throws RemoteException
```

A remote entity registers interest in events taking place within the session manager by calling this method. After registering the session manager can notify the entity via the Servlet's SessionObserver interface.

Parameters:

id - A unique string identifying the registering entity

Throws: RemoteException

if some communication failure occurs.

- **sessionIdList**

```
public abstract java.util.Vector sessionIdList() throws
RemoteException
```

Return an array containing all the current Session object identifiers.

Returns:

A Vector instance containing the Session object id's.

Throws: RemoteException

if some communication failure occurs.

- **shutdown**

```
public abstract void shutdown(String id) throws RemoteException
```

A remote entity informs the Session Manager that it is prepared to have the Manager shutdown operations. The Session Manager will shutdown operation as soon as it receives shutdown notifications from all registered entities.

Parameters:

id - A unique string identifying the entity

Throws: RemoteException

if some communication failure occurs.

- **unregister**

```
public abstract void unregister(String id) throws
RemoteException
```

A remote entity calls this method to stop further notifications being sent by the session manager.

Parameters:

id - A unique string identifying the entity.

Throws: RemoteException

if some communication failure occurs.

Interface
multiserv.sessionmgr.SessionNotification

public abstract interface SessionNotification
Provides the interface through which the Session Manager can transfer notification data. Any class which implements this interface must also implement interface java.io.Serializable.

Variable Index

- MANAGER_RELOAD
- MANAGER_RUNNING
- MANAGER_SHUTDOWN
- MANAGER_STOPPING
- SESSION_EXPIRATION

Method Index

- reason()
  Returns an integer code representing the reason for the notification.
- sessionId()
  Returns a string giving the session id for the Session object which caused the notification to be issued.
- sessionObj()
  Returns a reference to a copy of the Session object which this NotificationData instance relates to.

Variables

● MANAGER_RELOAD

```
public static final int MANAGER_RELOAD
```

● MANAGER_RUNNING

```
public static final int MANAGER_RUNNING
```

● MANAGER_SHUTDOWN

```
public static final int MANAGER_SHUTDOWN
```

● MANAGER_STOPPING

```
public static final int MANAGER_STOPPING
```

● SESSION_EXPIRATION

```
public static final int SESSION_EXPIRATION
```

Methods

- **reason**

```
public abstract int reason()
```

Returns an integer code representing the reason for the notification. In general, these integer codes will be application dependent.

- **sessionId**

```
public abstract java.lang.String sessionId()
```

Returns a string giving the session id for the Session object which caused the notification to be issued. This will return null if the notification is not connected with any Session object.

- **sessionObj**

```
public abstract multiserv.sessionmgr.Session sessionObj()
```

Returns a reference to a copy of the Session object which this NotificationData instance relates to. For example, if this is a session expiry notification then this method will return a reference to a copy of the Session object which was removed from the Session Manager.

Interface multiserv.sessionmgr.SessionObserver

public abstract interface **SessionObserver**
extends Remote
Provides the interface via which the Session Manager can notify remote entities.

Method Index

- notify(SessionNotification)
  Called by the Session Manager to notify the entities which implement this interface.

Methods

notify

```
public abstract void notify(SessionNotification nofn) throws
RemoteException
```

Called by the Session Manager to notify the entities which implement this interface.

Parameters:
nofn - Interface which accesses data relating to the notification.

Throws: RemoteException
if some communication failure occurs.

Interface multiserv.sessionmgr.SessionTags

public abstract interface SessionTags
Defines the constants used for basic sessions

Variable Index

- APPL_OPERATION
- BROWSER_TAG
- BROWSER_TOKEN
- COMMENT_TOKEN
- COMPUTER_TAG
- COMPUTER_TOKEN
- CONNECTION_TAG
- CONNECTION_TOKEN
- COOKIE_TAG
- DATE_TOKEN
- DOCNAME_TAG
- EMAIL_TAG
- END_TOKEN
- GUC_TOKEN
- LC_OPT_PREFIX
- LOGIN_TAG
- LOGOUT_OP
- MSG1_TOKEN
- MSG2_TOKEN
- NAME_TAG
- NAME_TOKEN
- OPT_PREFIX
- OP_TAG
- OP_TOKEN
- PASSWORD_TAG
- SESSIONID_TAG
- SESSIONID_TOKEN
- SESS_ACTIVE_TAG
- SESS_ID_TAG
- SESS_IP_ADDR_TAG
- SESS_START_TAG
- SESS_USER_ID_TAG
- SESS_USER_TAG
- SPEED_TAG
- SPEED_TOKEN
- START_TOKEN

- STATUS_TOKEN
- TERMINATE_TAG
- TIMEOUT_TAG
- TIMEOUT_TOKEN
- TIME_TOKEN
- TYPE_TAG
- TYPE_TOKEN
- URL_TAG
- URL_TOKEN
- USERID_TAG

  The following constants define the HTML tags which are used to identify data fields.

- USERNAME_TAG

Variables

- APPL_OPERATION

```
public static final java.lang.String APPL_OPERATION
```

- BROWSER_TAG

```
public static final java.lang.String BROWSER_TAG
```

- BROWSER_TOKEN

```
public static final java.lang.String BROWSER_TOKEN
```

- COMMENT_TOKEN

```
public static final java.lang.String COMMENT_TOKEN
```

- COMPUTER_TAG

```
public static final java.lang.String COMPUTER_TAG
```

- COMPUTER_TOKEN

```
public static final java.lang.String COMPUTER_TOKEN
```

- CONNECTION_TAG

```
public static final java.lang.String CONNECTION_TAG
```

- CONNECTION_TOKEN

```
public static final java.lang.String CONNECTION_TOKEN
```

- COOKIE_TAG

```
public static final java.lang.String COOKIE_TAG
```

- DATE_TOKEN

```
public static final java.lang.String DATE_TOKEN
```

- DOCNAME_TAG

```
public static final java.lang.String DOCNAME_TAG
```

●EMAIL_TAG

```
public static final java.lang.String EMAIL_TAG
```

●END_TOKEN

```
public static final java.lang.String END_TOKEN
```

●GUC_TOKEN

```
public static final java.lang.String GUC_TOKEN
```

●LC_OPT_PREFIX

```
public static final java.lang.String LC_OPT_PREFIX
```

●LOGIN_TAG

```
public static final java.lang.String LOGIN_TAG
```

●LOGOUT_OP

```
public static final java.lang.String LOGOUT_OP
```

●MSG1_TOKEN

```
public static final java.lang.String MSG1_TOKEN
```

●MSG2_TOKEN

```
public static final java.lang.String MSG2_TOKEN
```

●NAME_TAG

```
public static final java.lang.String NAME_TAG
```

●NAME_TOKEN

```
public static final java.lang.String NAME_TOKEN
```

●OPT_PREFIX

```
public static final java.lang.String OPT_PREFIX
```

●OP_TAG

```
public static final java.lang.String OP_TAG
```

●OP_TOKEN

```
public static final java.lang.String OP_TOKEN
```

●PASSWORD_TAG

```
public static final java.lang.String PASSWORD_TAG
```

●SESSIONID_TAG

```
public static final java.lang.String SESSIONID_TAG
```

●SESSIONID_TOKEN

```
public static final java.lang.String SESSIONID_TOKEN
```

SESS_ACTIVE_TAG

public static final java.lang.String SESS_ACTIVE_TAG
SESS_ID_TAG

public static final java.lang.String SESS_ID_TAG
- SESS_IP_ADDR_TAG

public static final java.lang.String SESS_IP_ADDR_TAG
- SESS_START_TAG

public static final java.lang.String SESS_START_TAG
- SESS_USER_ID_TAG

public static final java.lang.String SESS_USER_ID_TAG
- SESS_USER_TAG

public static final java.lang.String SESS_USER_TAG
- SPEED_TAG

public static final java.lang.String SPEED_TAG
- SPEED_TOKEN

public static final java.lang.String SPEED_TOKEN
- START_TOKEN

public static final java.lang.String START_TOKEN
- STATUS_TOKEN

public static final java.lang.String STATUS_TOKEN
- TERMINATE_TAG

public static final java.lang.String TERMINATE_TAG
- TIMEOUT_TAG

public static final java.lang.String TIMEOUT_TAG
- TIMEOUT_TOKEN

public static final java.lang.String TIMEOUT_TOKEN
- TIME_TOKEN

public static final java.lang.String TIME_TOKEN
- TYPE_TAG

public static final java.lang.String TYPE_TAG
- TYPE_TOKEN

public static final java.lang.String TYPE_TOKEN
- URL_TAG

public static final java.lang.String URL_TAG
URL_TOKEN

```
public static final java.lang.String URL_TOKEN
```

USERID_TAG

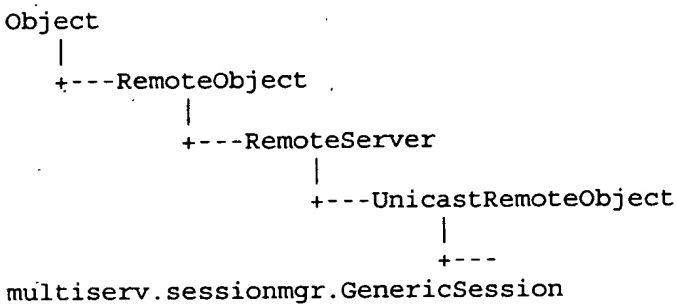
```
public static final java.lang.String USERID_TAG
```

The following constants define the HTML tags which are used to identify data fields. This file could possibly be generated automatically from or along with the HTML documents themselves.

- USERNAME_TAG

```
public static final java.lang.String USERNAME_TAG
```

Class multiserv.sessionmgr.GenericSession



public class **GenericSession**
extends UnicastRemoteObject
implements Session, Serializable, Cloneable

Variable Index

- expired
- lastAccessed

Constructor Index

- multiserv.sessionmgr.GenericSession()

Method Index

- expire()
  Mark this object as expired.
- getDouble(String)
  Get the value of a double precision floating point field within the associated SessionImpl instance.
- getInt(String)
  Get the value of an integer type field within the associated SessionImpl instance.
- getLastAccessed()
  Get the time at which this instance was last accessed.
- getLong(String)
  Get the value of a long integer field within the associated SessionImpl instance.
- getObject(String)
  Get the value of a field within the associated SessionImpl instance.
- hasExpired()
  Determine if this object has expired

- sessionFailure(Exception)
  General exception handler for the session object
- sessionFailure(String, Exception)
  General exception handler for the session object
- setDouble(String, double)
  Set the value of a double precision floating point field within the associated SessionImpl instance.
- setInt(String, int)
  Set the value of an integer type field within the associated SessionImpl instance.
- setLong(String, long)
  Set the value of a long integer field within the associated SessionImpl instance.
- setObject(String, Object)
  Set the value of a field within the associated SessionImpl instance.
- touch()
  Mark the time when this object was last accessed as now.

Variables

- expired

```
protected boolean expired
```

- lastAccessed

```
protected long lastAccessed
```

Constructors

- GenericSession

```
public GenericSession() throws RemoteException
```

Methods

- expire

```
public void expire() throws RemoteException
```

Mark this object as expired.

- getDouble

```
public double getDouble(String fieldName) throws
SessionAccessException
```

Get the value of a double precision floating point field within the associated SessionImpl instance.

Parameters:
fieldName - A string specifying the name of the field to set.
Returns:
The value of the field.
Throws: SessionAccessException

if there was a problem accessing the fields value.

getInt

```
public int getInt(String fieldName) throws
SessionAccessException
```

Get the value of an integer type field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **getLastAccessed**

```
public long getLastAccessed() throws RemoteException
```

Get the time at which this instance was last accessed.

Returns:

the time in milliseconds since EPOCH when this object was last accessed.

- **getLong**

```
public long getLong(String fieldName) throws
SessionAccessException
```

Get the value of a long integer field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **getObject**

```
public java.lang.Object getObject(String fieldName) throws
SessionAccessException
```

Get the value of a field within the associated SessionImpl instance. The field type must be an extension of Object and must also implement the Serializable interface.

Parameters:

fieldName - A string specifying the name of the field to set.

Returns:

The value of the field.

Throws: SessionAccessException

if there was a problem accessing the fields value.

hasExpired

```
public boolean hasExpired() throws RemoteException
```

Determine if this object has expired

Returns:

true if the object has expired, false otherwise.

- **sessionFailure**

```
protected void sessionFailure(Exception e) throws
SessionAccessException
```

General exception handler for the session object

Parameters:

e - the exception which is being handled

Throws: SessionAccessException

there was a problem while accessing a field.

- **sessionFailure**

```
protected void sessionFailure(String fieldName,
                              Exception e) throws
SessionAccessException
```

General exception handler for the session object

Parameters:

e - the exception which is being handled

Throws: SessionAccessException

there was a problem while accessing a field.

- **setDouble**

```
public void setDouble(String fieldName,
                      double value) throws
SessionAccessException
```

Set the value of a double precision floating point field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

value - The value to set the field to.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **setInt**

```
public void setInt(String fieldName,
                   int value) throws SessionAccessException
```

Set the value of an integer type field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

value - The value to set the field to.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **setLong**

```
public void setLong(String fieldName,
                    long value) throws SessionAccessException
```

Set the value of a long integer field within the associated SessionImpl instance.

Parameters:

fieldName - A string specifying the name of the field to set.

value - The value to set the field to.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **setObject**

```
public void setObject(String fieldName,
                      Object value) throws
SessionAccessException
```

Set the value of a field within the associated SessionImpl instance. The field type must be an extension of Object and must also implement the Serializable interface.

Parameters:

fieldName - A string specifying the name of the field to set.

value - The value to set the field to.

Throws: SessionAccessException

if there was a problem accessing the fields value.

- **touch**

```
public void touch() throws RemoteException
```

Mark the time when this object was last accessed as now.

Class multiserv.sessionmgr.NotificationData

```
Object
   |
   +---multiserv.sessionmgr.NotificationData
```

public class **NotificationData**
extends Object
implements SessionNotification, Serializable
Provides an implementation of the SessionNotification interface

Constructor Index

- multiserv.sessionmgr.NotificationData(int, String, Session)

Method Index

- reason()
  Returns an integer code representing the reason for the notification.
- sessionId()
  Returns a string giving the session id for the Session object which caused the notification to be issued.
- sessionObj()
  Returns a reference to a copy of the Session object which this NotificationData instance relates to.

Constructors

- **NotificationData**

```
public NotificationData(int code,
                        String id,
                        Session sess)
```

Methods

- **reason**

```
public int reason()
```

Returns an integer code representing the reason for the notification. In general, these integer codes will be application dependent.

- **sessionId**

```
public java.lang.String sessionId()
```

Returns a string giving the session id for the Session object which caused the notification to be issued. This will return null if the notification is not connected with any Session object.

sessionObj

```
public multiserv.sessionmgr.Session sessionObj()
```

Returns a reference to a copy of the Session object which this NotificationData instance relates to. For example, if this is a session expiry notification then this method will return a reference to a copy of the Session object which was removed from the Session Manager.

Class multiserv.sessionmgr.ObserverMap

```
Object
   |
   +---Dictionary
           |
           +---Hashtable
                   |
                   +---multiserv.sessionmgr.ObserverMap
```

public class **ObserverMap**
extends Hashtable
The ObserverMap class stores the currently actice SessionObserver instances within the session manager.

Constructor Index

- multiserv.sessionmgr.ObserverMap()
- multiserv.sessionmgr.ObserverMap(int)
- multiserv.sessionmgr.ObserverMap(int, float)

Method Index

- putObserver(String)
  Insert an observer object within the map using the remote entity id string as the key.
- removeObserver(String)
  Remove an observer object from the map.

Constructors

ObserverMap

```
public ObserverMap()
```

ObserverMap

```
public ObserverMap(int capacity)
```

ObserverMap

```
public ObserverMap(int capacity,
                   float loadFactor)
```

Methods

putObserver

```
public void putObserver(String id)
```

Insert an observer object within the map using the remote entity id string as the key.

Parameters:

obs - The observer object to insert in the map. It must be implement the SessionObserver interface.

id - The remote entity id string.

- **removeObserver**

```
public void removeObserver(String id)
```

Remove an observer object from the map.

Parameters:

id - The id string for the observer to remove

Class multiserv.sessionmgr.SessionExpirer

```
Object
   |
   +---Thread
           |
           +---multiserv.sessionmgr.SessionExpirer
```

public class **SessionExpirer**
extends Thread
A thread object which expires stale session objects within the session manager at regular intervals.

Constructor Index

- multiserv.sessionmgr.SessionExpirer(SessionMgrImpl)

Method Index

- haltExpiries()
- run()

Constructors

- SessionExpirer

```
public SessionExpirer(SessionMgrImpl mgr)
```

Methods

- **haltExpiries**

```
public void haltExpiries()
```

- **run**

```
public void run()
```

Overrides:
run in class Thread

Class multiserv.sessionmgr.SessionIdFactory

```
Object
   |
   +---multiserv.sessionmgr.SessionIdFactory
```

public class **SessionIdFactory**
extends Object

Constructor Index

- multiserv.sessionmgr.SessionIdFactory()

Method Index

- createSessionId()
  Creates a randomly generated string with NUMDIG digits, NUMLOWER lowercase characters, and NUMUPPER characters arranged randomly.

Constructors

- **SessionIdFactory**

```
public SessionIdFactory()
```

Methods

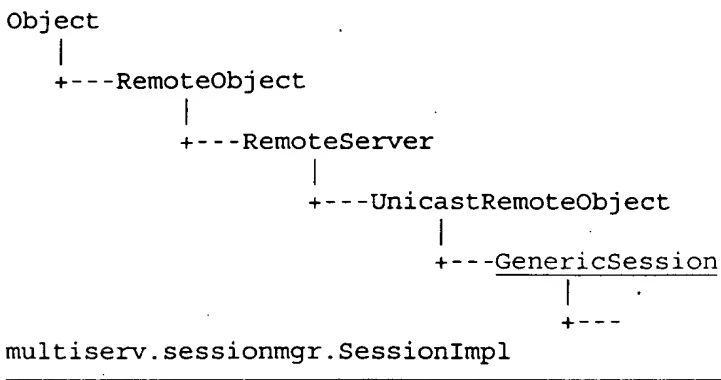
- **createSessionId**

```
public static java.lang.String createSessionId()
```

Creates a randomly generated string with NUMDIG digits, NUMLOWER lowercase characters, and NUMUPPER characters arranged randomly.

Returns:
the randomly generated string

Class multiserv.sessionmgr.SessionImpl



public class **SessionImpl**
extends GenericSession
implements SessionTags
Contains the application dependent data fields for the session.

Variable Index

- sess_active
- sess_ip_addr
- sess_start
- sess_user
  The actual Session Object members
- sess_user_id

Constructor Index

- multiserv.sessionmgr.SessionImpl()
  Get a new SessionImpl object within the session manager.
- multiserv.sessionmgr.SessionImpl(String, Long, BigDecimal, BigDecimal, String)
  Get a new SessionImpl object within the session manager.
- multiserv.sessionmgr.SessionImpl(Session)
  Get a new SessionImpl object within the session manager.

Method Index

- getSession()
  Get a new SessionImpl object within the session manager.
- getSession(String, Long, BigDecimal, BigDecimal, String)
  Get a new SessionImpl object within the session manager.
- getSession(Session)

Get a new SessionImpl object within the session manager.

Variables

sess_active

```
protected java.math.BigDecimal sess_active
```

sess_ip_addr

```
protected java.lang.String sess_ip_addr
```

sess_start

```
protected java.math.BigDecimal sess_start
```

sess_user

```
protected java.lang.String sess_user
```

The actual Session Object members

sess_user_id

```
protected java.lang.Long sess_user_id
```

Constructors

SessionImpl

```
public SessionImpl() throws RemoteException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

SessionImpl

```
public SessionImpl(String sess_user,
                   Long sess_user_id,
                   BigDecimal sess_start,
                   BigDecimal sess_active,
                   String sess_ip_addr) throws RemoteException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Parameters:

sess_user - The user
sess_user_id - The user id
sess_start - Time session was started
sess_active - Last time the session was active
sess_ip_addr - Remote IP Address

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

SessionImpl

```
public SessionImpl(Session initial) throws RemoteException,
SessionAccessException, NotSerializableException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.
Parameters:
initial - The initial session
Returns:
A string used to identify the new session in subsequent accesses to the session manager.

Methods

- **getSession**

```
public static multiserv.sessionmgr.SessionImpl getSession()
throws RemoteException, SessionAccessException,
NotSerializableException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

- **getSession**

```
public static multiserv.sessionmgr.SessionImpl getSession(String
sess_user, Long sess_user_id,
BigDecimal sess_start,BigDecimal sess_active,
String sess_ip_addr) throws RemoteException,
SessionAccessException, NotSerializableException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Parameters:

sess_user - The user name
sess_user_id - The user id
sess_start - Time session was started
sess_active - Last time the session was active
sess_ip_addr - Remote IP Address

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

- **getSession**

```
public static multiserv.sessionmgr.SessionImpl
getSession(Session initial) throws RemoteException,
SessionAccessException, NotSerializableException
```

Get a new SessionImpl object within the session manager. Gives the user the chance to override the default session behaviour.

Parameters:

session - Contains the initial session data.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

Class multiserv.sessionmgr.SessionMap

```
Object
   |
   +---Dictionary
           |
           +---Hashtable
                   |
                   +---multiserv.sessionmgr.SessionMap
```

public class **SessionMap**
extends Hashtable
The SessionMap class stores the currently active SessionImpl instances within the session manager.

Constructor Index

- multiserv.sessionmgr.SessionMap()
- multiserv.sessionmgr.SessionMap(int)
- multiserv.sessionmgr.SessionMap(int, float)

Method Index

- getSession(String)
  Retrieve a session object from within the using the specified session id key.
- putSession(Session, String)
  Insert a session object within the map using the session id string as the key.
- removeSession(String)
  Remove a session object from the map.

Constructors

SessionMap

```
public SessionMap()
```

SessionMap

```
public SessionMap(int capacity)
```

SessionMap

```
public SessionMap(int capacity,
                  float loadFactor)
```

Methods

getSession

```
public multiserv.sessionmgr.Session getSession(String sessionId)
```

Retrieve a session object from within the using the specified session id key.

Parameters:

sessionId - The session id string.

Returns:

The session object if the session id is valid otherwise null.

- **putSession**

```
public void putSession(Session session,
                       String sessionId)
```

Insert a session object within the map using the session id string as the key.

Parameters:

session - The session object to insert in the map. It must be a subclass of GenericSession.

sessionId - The session id string.

- **removeSession**

```
public void removeSession(String sessionId)
```

Remove a session object from the map.

Parameters:

sessionId - The id string for the session to remove

Class
multiserv.sessionmgr.SessionMgrConnection

```
Object
   |
   +---Thread
           |
           +---multiserv.sessionmgr.SessionMgrConnection
```

public class SessionMgrConnection
extends Thread

Constructor Index

- multiserv.sessionmgr.SessionMgrConnection(SessionMgrImpl, Socket)

Method Index

- run()

Constructors

- SessionMgrConnection

```
public SessionMgrConnection(SessionMgrImpl mgr,
                            Socket clientSocket)
```

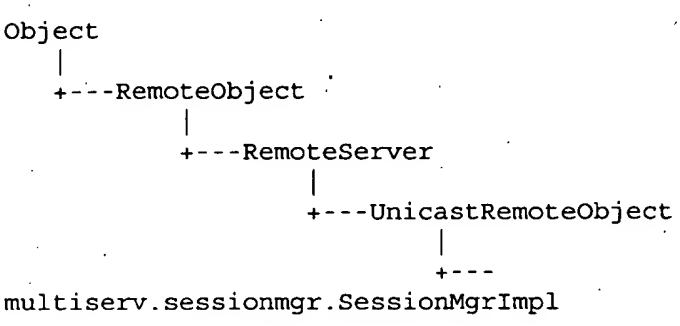
Methods

- run

```
public void run()
```

Overrides:
run in class Thread

Class multiserv.sessionmgr.SessionMgrImpl



public class **SessionMgrImpl**
extends UnicastRemoteObject
implements SessionMgr
The SessionMgr remote interface is used to control the session manager.

Variable Index

- config
  Contains the application configuration information

Constructor Index

- multiserv.sessionmgr.SessionMgrImpl(String, String, String)
  Construct a new instance of SessionMgrImpl.

Method Index

- closeSession(String)
  Remove an existing SessionImpl object from the session manager.
- expireOldSessions()
- finalize()
- initSession(Session)
  Create a new SessionImpl object within the session manager.
- log(String)
  Write information to stdout tagged with the date and time.
- main(String[])
  The main method for the session manager process.
- notifyServlets(int, String, Session)
- register(String)
  A remote entity registers interest in events taking place within the session manager by calling this method.
- reload()
- restoreState()

Restores the contents of the Session map from persistent store.

- saveState()

  Saves the contents of the Session map to persistent store.
- sessionIdList()

  Return an array containing all the current Session object identifiers.
- sessionObjName(String)

  Constructs the session object name given the session id.
- shutdown(String)

  A remote entity informs the Session Manager that it is prepared to have the Manager shutdown operations.
- shuttingdown()
- trace(String)
- unregister(String)

  A remote entity calls this method to stop further notifications being sent by the session manager.

Variables

- **config**

```
public multiserv.config.Config config
```

Contains the application configuration information

Constructors

- **SessionMgrImpl**

```
public SessionMgrImpl(String rmihost,
                      String name,
                      String configFile) throws RemoteException,
MalformedURLException
```

Construct a new instance of SessionMgrImpl.

Parameters:

managerName - Identifies this session manager instance

name - The name by which the session manager will be known

configFile - The configuration file

Methods

- **closeSession**

```
public void closeSession(String sessionId) throws
RemoteException, MalformedURLException, NotBoundException
```

Remove an existing SessionImpl object from the session manager.

Parameters:

sessionId - A string which identifies the session to be removed.

Throws: RemoteException

some communication problem occurred.

Throws: NotBoundException

the session which is being closed does not have it's interface bound to the RMI registry.

expireOldSessions

```
public void expireOldSessions()
```

- **finalize**

```
protected void finalize()
```

Overrides:

finalize in class Object

- **initSession**

```
public java.lang.String initSession(Session initial) throws
RemoteException, MalformedURLException, SessionAccessException,
NotSerializableException
```

Create a new SessionImpl object within the session manager.

Parameters:

session - Contains the initial session data.

Returns:

A string used to identify the new session in subsequent accesses to the session manager.

- **log**

```
public void log(String message)
```

Write information to stdout tagged with the date and time.

Parameters:

message - the information to output

- **main**

```
public static void main(String[] args)
```

The main method for the session manager process. This installs a security manager as well as creating a registry so that the clients can lookup the manager objects. The session manager must be started with a single command line argument. This argument specifies the session manager name.

- **notifyServlets**

```
protected void notifyServlets(int reason,
                              String sid,
                              Session sess) throws
RemoteException
```

- **register**

```
public void register(String id) throws RemoteException
```

A remote entity registers interest in events taking place within the session manager by calling this method. After this the session manager can notify the entity via the Servlet's SessionObserver interface.

Parameters:

id - A unique string identifying the registering entity. The name should be an RMI addressable ID. That is, you should be able to tack "rmi://" on to the front of it. Suggestion is "host/servletId"
Throws: RemoteException
if some communication failure occurs.

- **reload**

```
public void reload() throws RemoteException
```

- **restoreState**

```
protected void restoreState()
```

Restores the contents of the Session map from persistent store.

- **saveState**

```
protected void saveState()
```

Saves the contents of the Session map to persistent store.

- **sessionIdList**

```
public java.util.Vector sessionIdList() throws RemoteException
```

Return an array containing all the current Session object identifiers.
Returns:
An Vector instance containing the Session object id's.
Throws: RemoteException
if some communication failure occurs.

- **sessionObjName**

```
protected java.lang.String sessionObjName(String sid)
```

Constructs the session object name given the session id.
Parameters:
sid - the session id string for the object
Returns:
a string containing the session object name or the empty string if sid is null.

- **shutdown**

```
public void shutdown(String id) throws RemoteException
```

A remote entity informs the Session Manager that it is prepared to have the Manager shutdown operations. The Session Manager will shutdown operation as soon as it receives shutdown notifications from all registered entities.
Parameters:
id - A unique string identifying the entity
Throws: RemoteException
if some communication failure occurs.

- **shuttingdown**

```
public void shuttingdown()
```

trace

```
public void trace(String message)
```

- **unregister**

```
public void unregister(String id) throws RemoteException
```

A remote entity calls this method to stop further notifications being sent by the session manager.

Parameters:

id - A unique string identifying the entity.

Throws: RemoteException

if some communication failure occurs.

See Also:

register

Class multiserv.sessionmgr.SessionMgrShutdown

```
Object
   |
   +---Thread
           |
           +---multiserv.sessionmgr.SessionMgrShutdown
```

public class SessionMgrShutdown
extends Thread
A thread object which shuts down the SessionMgr process

Constructor Index

- multiserv.sessionmgr.SessionMgrShutdown()

Method Index

- run()

Constructors

- SessionMgrShutdown

```
public SessionMgrShutdown()
```

Methods

- run

```
public void run()
```

Overrides:
run in class Thread

Class multiserv.sessionmgr.SocketHandler

```
Object
    |
    +---Thread
            |
            +---multiserv.sessionmgr.SocketHandler
```

public class **SocketHandler**
extends Thread
A thread object which listens on a socket for connection attempts and then services commands from authorized clients. Normally only the host on which the session manager is running or the loopback device are allowed. The property multiserv.sessionmgr.allowedIPs of IPAddresses allows other hosts access - it should coonsist of white space separated IP addresses.

Constructor Index

- multiserv.sessionmgr.SocketHandler(SessionMgrImpl)
  Create a socket handler

Method Index

- halt()
- run()

Constructors

- SocketHandler

```
public SocketHandler(SessionMgrImpl mgr)
```

Create a socket handler
Parameters:
mgr - Reference to the session manager instance

Methods

- **halt**

```
public void halt()
```

- **run**

```
public void run()
```

Overrides:
run in class Thread

Class
multiserv.sessionmgr.SessionAccessException

```
Object
   |
   +---Throwable
           |
           +---Exception
                   |
                   +---
multiserv.sessionmgr.SessionAccessException
```

public class **SessionAccessException**
extends Exception
This exception is thrown to indicate a problem with accessing data within a session object.

Constructor Index

- multiserv.sessionmgr.SessionAccessException(String)
  Constructs a new SessionAccessException with the specified descriptive message.

Constructors

- **SessionAccessException**

```
public SessionAccessException(String msg)
```

Constructs a new SessionAccessException with the specified descriptive message.

package multiserv.util

Interface Index

- Watchable

Class Index

- Cache
- FileCache
- FileCacheFactory
- FileSuffixFilter
- HTMLCache
- HTMLCacheFactory
- HTMLDocument
- Logger
- SendMail
- SoundEx
- TimedCounter
- Watcher

Exception Index

- CacheException

Interface multiserv.util.Watchable

public abstract interface Watchable

Method Index

Methods

●wakeup

```
public abstract void wakeup()
```

●watch

```
public abstract void watch()
```

Class multiserv.util.Cache

```
Object
   |
   +---multiserv.util.Cache
```

public abstract class Cache
extends Object
implements Watchable

Variable Index

- DEBUG

Constructor Index

- multiserv.util.Cache()
- multiserv.util.Cache(long)

Method Index

- addObject(String, Object, long)
- getKeys()
  Get the keys of the table rows stored in the Cache.
- getObject(String)
- interruptWatching()
- populate()
- reinitialize()
- removeAll()
- removeObject(String)
- repopulate()
- setWatcherDelay(long)
- startWatching()
- stopWatching()
- updateObject(String, Object, long)
- wakeup()
- watch()

Variables

- DEBUG

```
public static boolean DEBUG
```

Constructors

- Cache

```
public Cache()
```
- Cache

```
public Cache(long secs)
```

Methods

- addObject

```
public synchronized void addObject(String name,
                                   Object obj,
                                   long lastModified)
```

- getKeys

```
protected synchronized java.util.Enumeration getKeys()
```

Get the keys of the table rows stored in the Cache.

Returns:

An Enumeration of the available keys. The keys are stored as strings.

- getObject

```
public synchronized java.lang.Object getObject(String name)
throws CacheException
```

- interruptWatching

```
public void interruptWatching()
```

- populate

```
public abstract void populate()
```

- reinitialize

```
public abstract void reinitialize()
```

- removeAll

```
public synchronized void removeAll()
```

- removeObject

```
public synchronized void removeObject(String name)
```

- repopulate

```
public abstract void repopulate()
```

- setWatcherDelay

```
public void setWatcherDelay(long secs)
```

- startWatching

```
public void startWatching()
```

- stopWatching

```
public void stopWatching()
```

- updateObject

```
public synchronized void updateObject(String name,
                                      Object obj,
                                      long lastModified)
```

● wakeup

```
public final synchronized void wakeup()
```

● watch

```
public final synchronized void watch()
```

Class multiserv.util.FileCache

```
Object
   |
   +---Cache
          |
          +---multiserv.util.FileCache
```

public class FileCache
extends Cache

Method Index

- getDocument(String)
- main(String[])
- populate()
- reinitialize()
- repopulate()
- setSuffixes(String[])

Methods

getDocument

```
public java.lang.String getDocument(String name) throws
CacheException
```

main

```
public static void main(String[] args)
```

populate

```
public void populate()
```

Overrides:
populate in class Cache

reinitialize

```
public void reinitialize()
```

Overrides:
reinitialize in class Cache

repopulate

```
public void repopulate()
```

Overrides:
repopulate in class Cache

setSuffixes

```
public void setSuffixes(String[] fileSuffixes)
```

Class multiserv.util.FileCacheFactory

```
Object
   |
   +---multiserv.util.FileCacheFactory
```

public class FileCacheFactory
extends Object

Constructor Index

- multiserv.util.FileCacheFactory()

Method Index

- getCache(String)
- getCache(String, String, long, String[])

Constructors

- FileCacheFactory

```
public FileCacheFactory()
```

Methods

- getCache

```
public static multiserv.util.FileCache getCache(String
cacheName) throws IOException
```

- getCache

```
public static multiserv.util.FileCache getCache(String
cacheName,
                                                String dir,
                                                long
cachePeriod,
                                                String[]
suffixes) throws IOException
```

Class multiserv.util.FileSuffixFilter

```
Object
   |
   +---multiserv.util.FileSuffixFilter
```

public class **FileSuffixFilter**
extends Object
implements FilenameFilter

Constructor Index

- multiserv.util.FileSuffixFilter(String[])
- multiserv.util.FileSuffixFilter(String[], long)

Method Index

- accept(File, String)

Constructors

- **FileSuffixFilter**

```
public FileSuffixFilter(String[] suffixes)
```

- **FileSuffixFilter**

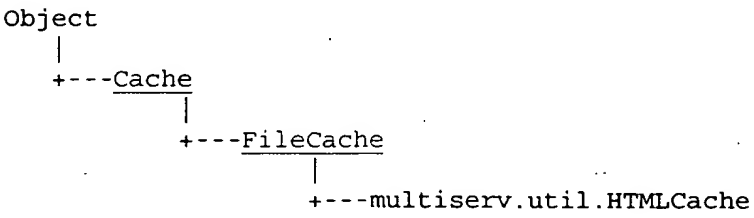
```
public FileSuffixFilter(String[] suffixes,
                        long modifiedSince)
```

Methods

- **accept**

```
public boolean accept(File dir,
                      String name)
```

Class multiserv.util.HTMLCache



public class **HTMLCache**
extends FileCache

Class multiserv.util.HTMLCacheFactory

```
Object
   |
   +---FileCacheFactory
          |
          +---multiserv.util.HTMLCacheFactory
```

public class HTMLCacheFactory
extends FileCacheFactory

Constructor Index

- multiserv.util.HTMLCacheFactory()

Method Index

- getCache(String, String, long, String[], Hashtable)

Constructors

- HTMLCacheFactory

```
public HTMLCacheFactory()
```

Methods

- getCache

```
public static multiserv.util.HTMLCache getCache(String
cacheName,
                                                String dir,
                                                long
cachePeriod,
                                                String[]
suffixes,
                                                Hashtable
tokens) throws IOException
```

Class multiserv.util.HTMLDocument

```
Object
   |
   +----multiserv.util.HTMLDocument
```

public class HTMLDocument
extends Object

Constructor Index

- multiserv.util.HTMLDocument(String, String)
- multiserv.util.HTMLDocument(String, String, Hashtable)
- multiserv.util.HTMLDocument(String, String, Hashtable, Hashtable, boolean)

Method Index

- buildListTokens(String, String, String, Vector, Hashtable, boolean)
- buildSimpleTokens(Hashtable, Hashtable)
  Convenience method which builds a token table to be passed to sendParseDocument().
- buildTokens(Hashtable, Hashtable, boolean)
  Convenience method which builds a token table to be passed to sendParseDocument().
- toString()

Constructors

HTMLDocument

```
public HTMLDocument(String cacheName,
                    String docName) throws IOException
```

HTMLDocument

```
public HTMLDocument(String cacheName,
                    String docName,
                    Hashtable tokens) throws IOException
```

HTMLDocument

```
public HTMLDocument(String cacheName,
                    String docName,
                    Hashtable tokens,
                    Hashtable tokenData,
                    boolean checkedNameVal) throws IOException
```

Methods

buildListTokens

```
public static void buildListTokens(String cacheName,
                                   String listDocToken,
                                   String listDocName,
                                   Vector data,
                                   Hashtable tokens,
                                   boolean checkedNameVal)
throws IOException
```

- **buildSimpleTokens**

```
public static void buildSimpleTokens(Hashtable data,
                                     Hashtable tokens)
```

Convenience method which builds a token table to be passed to sendParseDocument(). It basically goes through the passed Hashtable and builds another Hashtable where the keys are renamed as ~*key*~ That is, '~*' is tacked on the beginning and '*~' is tacked on the end of each key. This method could be useful for taking the data from a database query returned in a hashtable and building a token table to populate an HTML page.

Parameters:

data - The incoming Hashtable

tokens - The token table to be built. It might already contain some tokens - which will added to.

- **buildTokens**

```
public static void buildTokens(Hashtable data,
                               Hashtable tokens,
                               boolean checkedNameVal)
```

Convenience method which builds a token table to be passed to sendParseDocument(). It basically goes through the passed Hashtable and builds another Hashtable where the keys are renamed as ~*key*~ That is, '~*' is tacked on the beginning and '*~' is tacked on the end of each key. Special cases exist for keys starting with pvt or opt. These are treated as the special token types for checkboxes/radio buttons and selection lists respectively. This method could be useful for taking the data from a database query returned in a hashtable and building a token table to populate an HTML page.

Parameters:

data - The incoming Hashtable

tokens - The token table to be built. It might already contain some tokens - which will added to.

checkedNameVal - If checked, build PVT tokens as @# name=value# @ else just as @# name# @

- **toString**

```
public java.lang.String toString()
```

Overrides:

Class multiserv.util.Logger

```
Object
   |
   +---Thread
           |
           +---multiserv.util.Logger
```

public class **Logger**
extends Thread
A class which can be used for logging.

Constructor Index

- multiserv.util.Logger(String)
  Constructor for the Logger

Method Index

- getWriter()
- main(String[])
- run()

Constructors

- Logger

```
public Logger(String logFile) throws IOException
```

Constructor for the Logger
Parameters:
logFile - The file to which to log

Methods

- **getWriter**

```
public static java.io.PrintWriter getWriter() throws IOException
```

- **main**

```
public static void main(String[] argv)
```

- **run**

```
public void run()
```

Overrides:
run in class Thread

Class multiserv.util.SendMail

```
Object
   |
   +---multiserv.util.SendMail
```

public class SendMail
extends Object
implements Runnable
This is an interim class for sending mail. Sun hava a javamail API which is still in Beta Test and will be available for platform independent sending of mail. I did download it to check it out but it required another piece of Beta software so I thought I would leave it for now.

Constructor Index

- multiserv.util.SendMail(String, String, String, String, String, String)
- multiserv.util.SendMail(String, String, String, String, String, String, boolean)

Method Index

- main(String[])
- run()

Constructors

- SendMail

```
public SendMail(String from,
                String to,
                String cc,
                String replyTo,
                String subject,
                String message) throws IOException
```

- SendMail

```
public SendMail(String from,
                String to,
                String cc,
                String replyTo,
                String subject,
                String message,
                boolean mimed) throws IOException
```

Methods

main

```
public static void main(String[] argv)
  run

public void run()
```

Class multiserv.util.SoundEx

```
Object
   |
   +---multiserv.util.SoundEx
```

public class SoundEx
extends Object

Constructor Index

- multiserv.util.SoundEx()
- multiserv.util.SoundEx(String)

Method Index

- Calculate(String)
  Generates a soundex code for the input string.

Constructors

- SoundEx

```
public SoundEx()
```

- SoundEx

```
public SoundEx(String inputString)
```

Methods

- Calculate

```
public static java.lang.String Calculate(String inputString)
```

Generates a soundex code for the input string.

Parameters:

inputString - The string that is used to generate the soundex code

Returns:

A string representing the soundex code. If a code cannot be generated then the string "XXXX" is returned.

Class multiserv.util.Watcher

```
Object
   |
   +---multiserv.util.Watcher
```

public class Watcher
extends Object
implements Runnable

Constructor Index

- multiserv.util.Watcher(Watchable, long)

Method Index

- interrupt()
- run()
- start()
- stop()

Constructors

- Watcher

```
public Watcher(Watchable babe,
               long secs)
```

Methods

- interrupt

```
public void interrupt()
```

- run

```
public void run()
```

- start

```
public void start()
```

- stop

```
public void stop()
```

Class multiserv.util.CacheException

```
Object
   |
   +---Throwable
           |
           +---Exception
                   |
                   +---IOException
                           |
                           +---multiserv.util.CacheException
```

public class CacheException
extends IOException

Constructor Index

- multiserv.util.CacheException()
- multiserv.util.CacheException(String)
- multiserv.util.CacheException(String, Exception)

Method Index

- getCause()

Constructors

- CacheException

```
public CacheException()
```

- CacheException

```
public CacheException(String reason)
```

- CacheException

```
public CacheException(String reason,
                      Exception e)
```

Methods

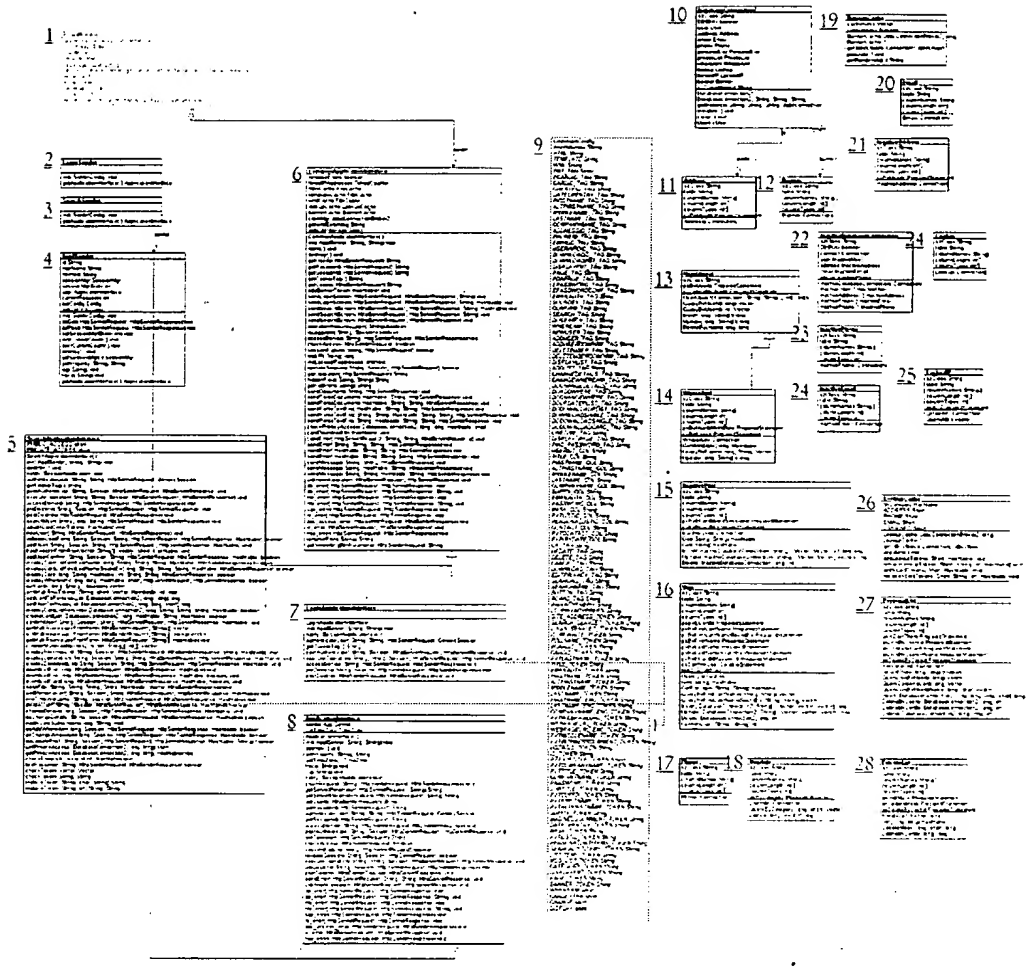
- getCause

```
public java.lang.Exception getCause()
```

UML Diagrams

The following UML diagrams provide a visual description of the interactions between the Java classes used by Ecardfile.

From the high level diagram you can click on the number by the side of the class and be brought to the large scale diagram.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

EcardNotifier

EcardNotifier
-parent:CommonApplicationInterface -myThread:Thread -cardId:long -eCardId:String -messageCache:String
+EcardNotifier(CommonApplicationInterface, String, long, String){constructor} +run( ):void +start( ):void +interrupt( ):void +stop( ):void +notifyUser(Hashtable, long, long, String, Hashtable):void

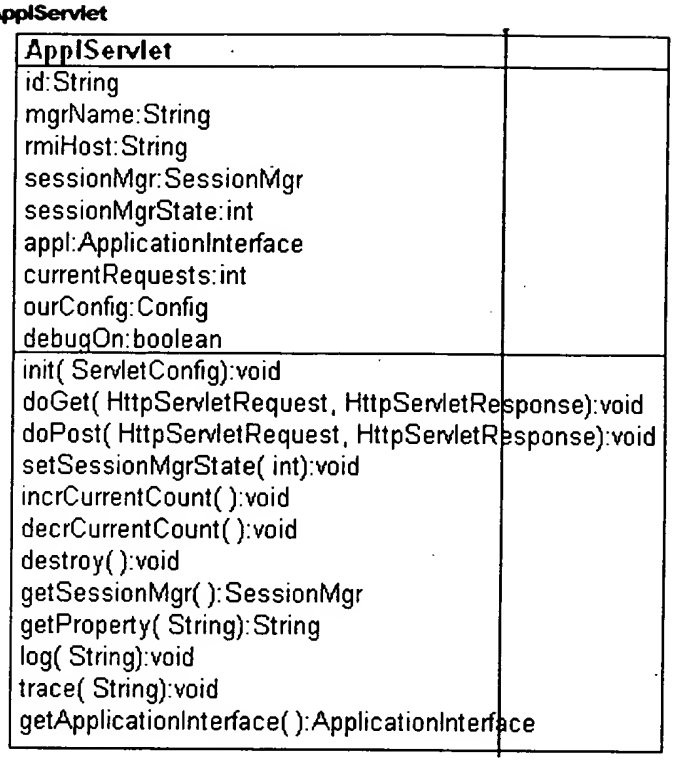
LoginServlet

LoginServlet
init(ServletConfig):void getApplicationInterface( ):ApplicationInterface

SearchServlet

SearchServlet
init(ServletConfig):void getApplicationInterface( ):ApplicationInterface

ApplServlet



SearchApplicationInterface

SearchApplicationInterface
PUBLIC_ACCESS:short
PRIVATE_ACCESS:short
SearchApplicationInterface( )
init(ApplServlet, String, String):void
destroy( ):void
notify(SessionNotification):void
authenticateUser(String, String, HttpServletRequest):GenericSession
getCookieTag( ):String
postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse):void
executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse):void
accessDenied(String, HttpServletRequest, HttpServletResponse):void
preDestroy(String, Session, HttpServletRequest, HttpServletResponse):void
postDestroy(HttpServletRequest, HttpServletResponse):void
searchUser(String, long, String, HttpServletRequest, HttpServletResponse):void
adjustCardDetailTokens(Hashtable):void
newUser(String, HttpServletRequest, HttpServletResponse):void
validateUserForm(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable):
addUser(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable):void
hashVectorFromRowVector(String[], Vector, short, Hashtable):void
addUserConfirm(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable):bo
displayCardListFirstLastName(long, String, String, String, Hashtable, HttpServletResponse):boole
displayCardListFirstLastNameSoundEx(long, String, String, String, Hashtable, HttpServletRespor
displayCard(long, String, Hashtable, int, String, String, HttpServletResponse):boolean
displayWhereAml(long, long, Hashtable, short, HttpServletRequest, HttpServletResponse):boolea
getCards(long, String[], boolean):Vector
addMultiRowTokens(String, short, Vector, Hashtable, int):void
addCardToPersonalList(DatabaseConnection2, long, long):long
addUserToPrivateList(DatabaseConnection2, long, long, short):long
displayCardListPersonal(DatabaseConnection2, long, String, String, String, Hashtable):boolean
createCardList(DatabaseConnection2, long, Hashtable, String, Hashtable):boolean
confirmUser(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable):void
getMultiRowValuesFromForm(HttpServletRequest, String[]):Vector
getMultiRowValuesFromForm(HttpServletRequest, String[], String):Vector
getMultiRowValuesFromForm(HttpServletRequest, String[], Hashtable):void
convertMultiRowHashes(Vector, String[], int[]):Vector
displayPersonalList(String, Session, HttpServletRequest, HttpServletResponse, String, Hashtable
displayUpdateList(String, Session, String, String, HttpServletRequest, HttpServletResponse, Has
displayDownloadList(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtabl
downloadFile(long, HttpServletRequest, HttpServletResponse, Hashtable):void
downloadFile(long, HttpServletRequest, HttpServletResponse, Hashtable, boolean):void
downloadCard(long, HttpServletRequest, HttpServletResponse, Hashtable, boolean):void
sendFile(String, String, String, String, Hashtable, Vector, HttpServletResponse):void
addPersonalList(String, Session, String, HttpServletRequest, HttpServletResponse, Hashtable):voi
oneTimeWelcome(String, Session, HttpServletRequest, HttpServletResponse, Hashtable):void
deleteUser(String, Session, HttpServletRequest, HttpServletResponse, Hashtable):boolean
changeDetails(long, Session, HttpServletRequest, HttpServletResponse, Hashtable):void
doChangeDetails(String, Session, HttpServletRequest, HttpServletResponse, Hashtable):boolean
notifyCardSubscribers(long, String):void
doAddWhereAml(long, Session, HttpServletRequest, HttpServletResponse, Hashtable):boolean
doChangeWhereAml(long, Session, HttpServletRequest, HttpServletResponse, Hashtable):boolea
doUpdatePlist(String, Session, HttpServletRequest, HttpServletResponse, Hashtable, String):boo
getPrivacyAccess(DatabaseConnection2, long, long):short
getPrivacyAccess(DatabaseConnection2, long, long, Hashtable):void
checkPrivacyAccess(short, Hashtable):void
findPassword(String, HttpServletRequest, HttpServletResponse):boolean
checkToken(Vector):Vector
checkToken(String):String
replaceToken(String, char, String):String
replaceToken(String, int, String):String

CommonApplicationInterface

CommonApplicationInterface
verboseErrors:boolean failedIPAddresses:TimedCounter htmlCache:FileCache templateCache:FileCache wmlCache:FileCache lookupCache:LookupCache bannerCache:BannerCache connMgr:JdbcConnectionBroker2 generalErrorMsg:String defaultPdaPage:String
CommonApplicationInterface( ) init(ApplServlet, String, String):void reinit( ):void destroy( ):void getUserId(HttpServletRequest):String getPassword(HttpServletRequest):String getSessionId(HttpServletRequest):String getCookieTag( ):String getCookie(HttpServletRequest):String addBannerToken(Hashtable):void sendLoginScreen(HttpServletRequest, HttpServletResponse, String):void sendLoginScreen(HttpServletRequest, HttpServletResponse, String, Hashtable):void sendSearchScreen(HttpServletRequest, HttpServletResponse, String, Hashtable):void sendSearchScreen(HttpServletRequest, HttpServletResponse, String):void sendSearchScreen(HttpServletRequest, HttpServletResponse):void operationRequiresLogin(String):boolean isLoggedIn(String, Session):boolean accessDenied(String, HttpServletRequest, HttpServletResponse):void checkAccess(HttpServletRequest):boolean sessionFailure(String, HttpServletRequest):boolean lockIP(String):long initLockedIPaddresses(int):void validateSession(String, Session, HttpServletRequest):boolean getOperation(HttpServletRequest):String hiddenField(String, String):String getFullPath(String):String sendDocument(String, HttpServletResponse):void sendParseDocument(Hashtable, String, HttpServletResponse):void sendParseDocument(Hashtable, String, String, HttpServletResponse):void sendParseTextFile(Hashtable, String, String, HttpServletResponse):void sendParseTextFile(String, String, Hashtable, String, String, HttpServletResponse):void sendParseTextFile(String, Hashtable, String, String, HttpServletResponse):void checkPrivacyAccess(DatabaseConnection2, long, long, Hashtable):short cardDownloadUrl(Hashtable):void sendError(HttpServletRequest, String, String, HttpServletResponse):void sendError(String, String, String, HttpServletResponse):void sendError(Hashtable, String, String, HttpServletResponse):void sendError(Hashtable, String, String, String, HttpServletResponse):void sendMessage(String, String, HttpServletResponse):void sendMessage(String, Hashtable, String, HttpServletResponse):void sendMessage(String, String, Hashtable, String, HttpServletResponse):void unknownOperation(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse, String):void nfe_error(HttpServletRequest, HttpServletResponse):void nfe_error(HttpServletRequest, HttpServletResponse, String):void sae_error(HttpServletRequest, HttpServletResponse):void re_error(HttpServletRequest, HttpServletResponse):void doc_access_error(HttpServletRequest, HttpServletResponse):void io_error(HttpServletRequest, HttpServletResponse):void nse_error(HttpServletRequest, HttpServletResponse):void verboseError(String):String getServletImgBtnParameter(HttpServletRequest):String

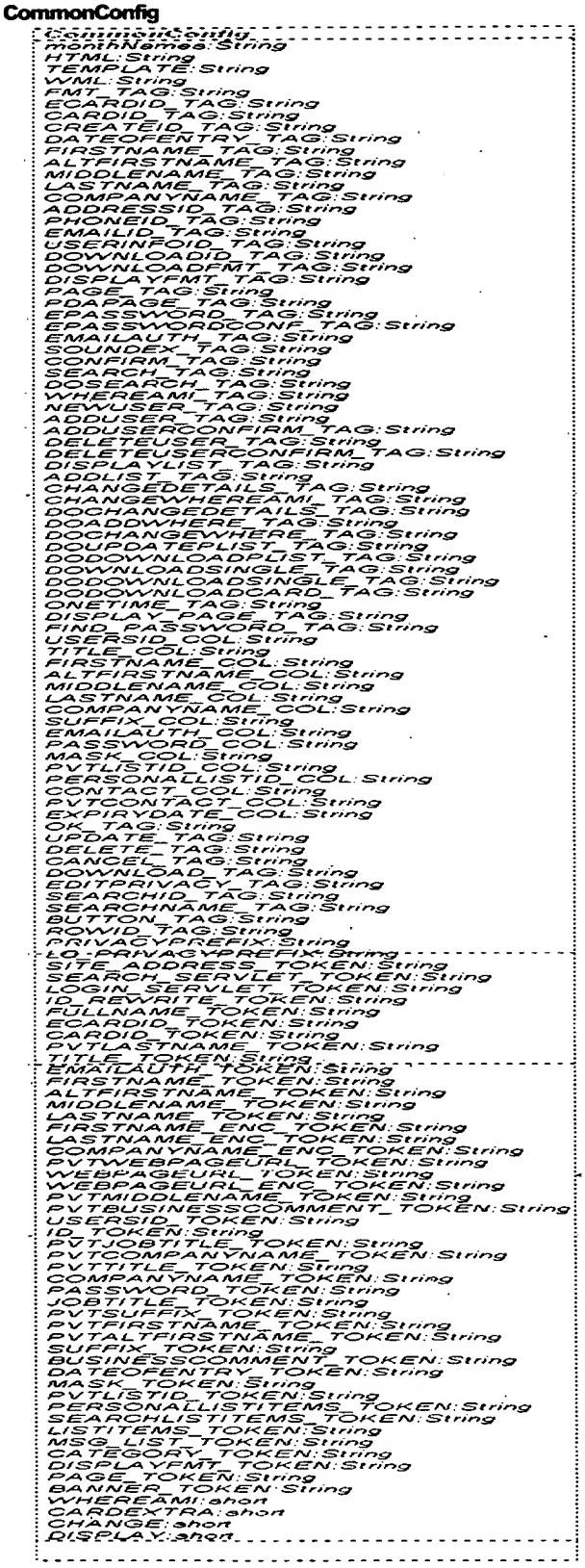
LoginApplicationInterface

LoginApplicationInterface
LoginApplicationInterface( ) init(ApplServlet, String, String):void notify(SessionNotification):void authenticateUser(String, String, HttpServletRequest):GenericSession getCookieTag( ):String postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse):void executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse):void accessDenied(String, HttpServletRequest, HttpServletResponse):void preDestroy(String, Session, HttpServletRequest, HttpServletResponse):void postDestroy(HttpServletRequest, HttpServletResponse):void

ApplicationInterface

ApplicationInterface
servlet:ApplServlet ourTimeZone:TimeZone
ApplicationInterface( ) ~~init(ApplServlet, String, String):void~~ destroy( ):void getProperty(String):String getTimeZone( ):TimeZone trace(String):void log(String):void notify(SessionNotification):void chainRequest(String, HttpServletRequest, HttpServletResponse):void getServletParameter(HttpServletRequest, String):String getServletParameterValues(HttpServletRequest, String):String[] getUserId(HttpServletRequest):String getPassword(HttpServletRequest):String authenticateUser(String, String, HttpServletRequest):GenericSession getSessionId(HttpServletRequest):String accessDenied(String, HttpServletRequest, HttpServletResponse):void postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse):void getOperation(HttpServletRequest):String checkAccess(HttpServletRequest):boolean sessionFailure(String, HttpServletRequest):boolean validateSession(String, Session, HttpServletRequest):boolean executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse):void preDestroy(String, Session, HttpServletRequest, HttpServletResponse):void postDestroy(HttpServletRequest, HttpServletResponse):void sendError(HttpServletRequest, String, String, HttpServletResponse):void unknownOperation(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse, String):void nfe_error(HttpServletRequest, HttpServletResponse):void nfe_error(HttpServletRequest, HttpServletResponse, String):void sae_error(HttpServletRequest, HttpServletResponse):void re_error(HttpServletRequest, HttpServletResponse):void doc_access_error(HttpServletRequest, HttpServletResponse):void io_error(HttpServletRequest, HttpServletResponse):void nse_error(HttpServletRequest, HttpServletResponse):void

CommonConfig



DatabaseConnection2

DatabaseConnection2
szClass:String DEBUG:boolean user:User address:Address email:Email phone:Phone personalList:PersonalList privateList:PrivateList whereAml:WhereAml lookup:Lookup lockedIP:LockedIP banner:Banner szClassMethod:String
DatabaseConnection2( ) DatabaseConnection2(String, String, String) getInstance(String, String, String):JdbcConnection Initialize( ):void close( ):void User( ):User

Address

Address
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psByUserId:PreparedStatement
Address(Connection)

Banner

Banner
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[]
Banner(Connection)

UserObject

UserObject
szClass:String psByUserId:PreparedStatement psDeleteByUserId:PreparedStatement
UserObject(Connection, String, String[], int[], int[]) QueryByUserId(long):Vector QueryByUserId(int):Vector Insert(long, String[]):long Update(long, String[]):long DeleteByUserId(long):long

WhereAml

WhereAml
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psByUserIdWithDate:PreparedStatement psByExpiry:PreparedStatement
WhereAml(Connection) GetWithDate(long):Hashtable QueryByExpiry(long, String):Vector Update(long, String[]):long

InactiveUser

InactiveUser
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psByECardIdAndSessionId:PreparedStatement psByECardId:PreparedStatement
InactiveUser(Connection) Get(String, String):Hashtable Get(String):Hashtable Insert(InactiveDatabaseConnection, String[], Vector, Vector, Vector):long Update(InactiveDatabaseConnection, String[], Vector, Vector, Vector):long Delete(InactiveDatabaseConnection, long):int

User

User
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psByECardId:PreparedStatement psByFirstLastName:PreparedStatement psByFirstLastNameSoundEx:PreparedStatement psByFirstName:PreparedStatement psByLastName:PreparedStatement psByLastNameSoundEx:PreparedStatement psByECardIdPassword:PreparedStatement csConfirmUser:CallableStatement MAX_ROWS:int
User(Connection) Get(String):Hashtable GetForLogin(String, String):Hashtable QueryByFirstLastName(String, String):Vector QueryByFirstLastNameSoundEx(String, String):Vector Insert(DatabaseConnection2, String[], Vector, Vector, Vector):long Update(DatabaseConnection2, String[], Vector, Vector, Vector):long Delete(DatabaseConnection2, long):int ConfirmUser(String, String):int

Phone

Phone
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[]
Phone(Connection)

UserInfo

UserInfo
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psByCategory:PreparedStatement
UserInfo(Connection) QueryByCategory(long, short):Vector Update(long, Vector):long

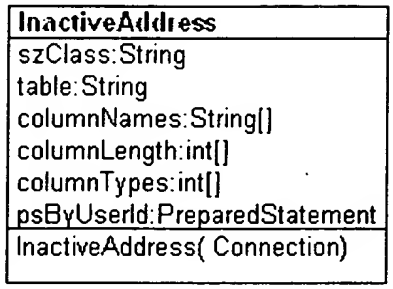
BannerCache

BannerCache
cacheKeys:Vector randomness:Random
BannerCache(JdbcConnectionBroker2, long) BannerCache( ) getJdbcObject(Connection):JdbcObject populate( ):void getRandomAd( ):String

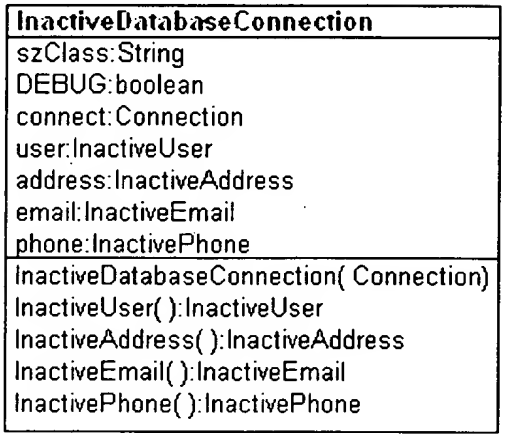
EMail

Email
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[]
Email(Connection)

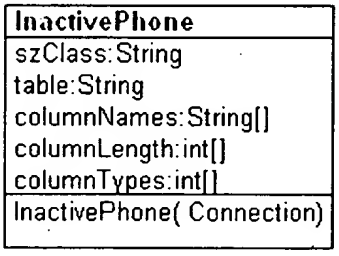
InactiveAddress



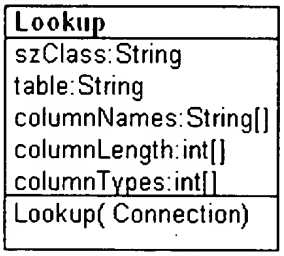
InactiveDatabaseConnection



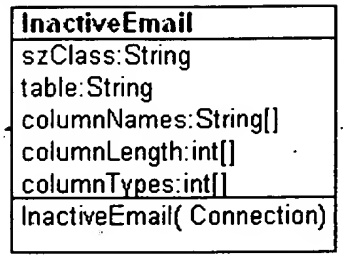
InactivePhone



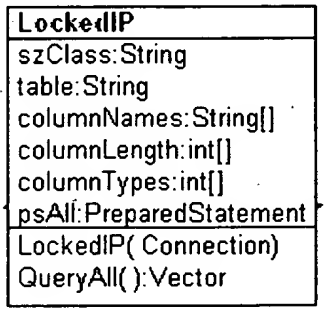
Lookup



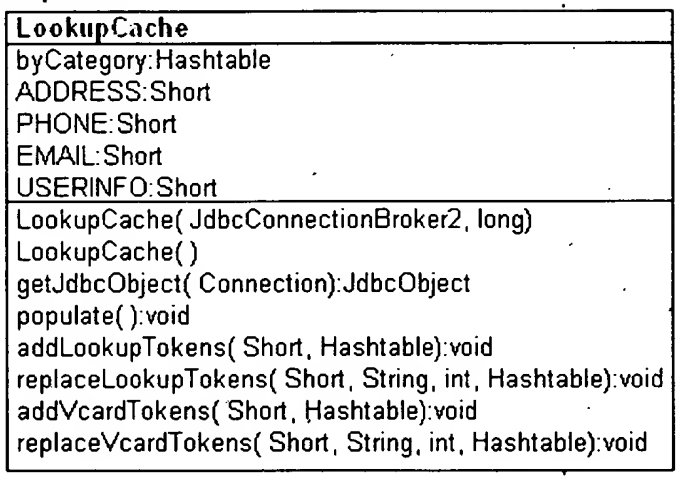
InactiveEmail



LockedIP



LookupCache

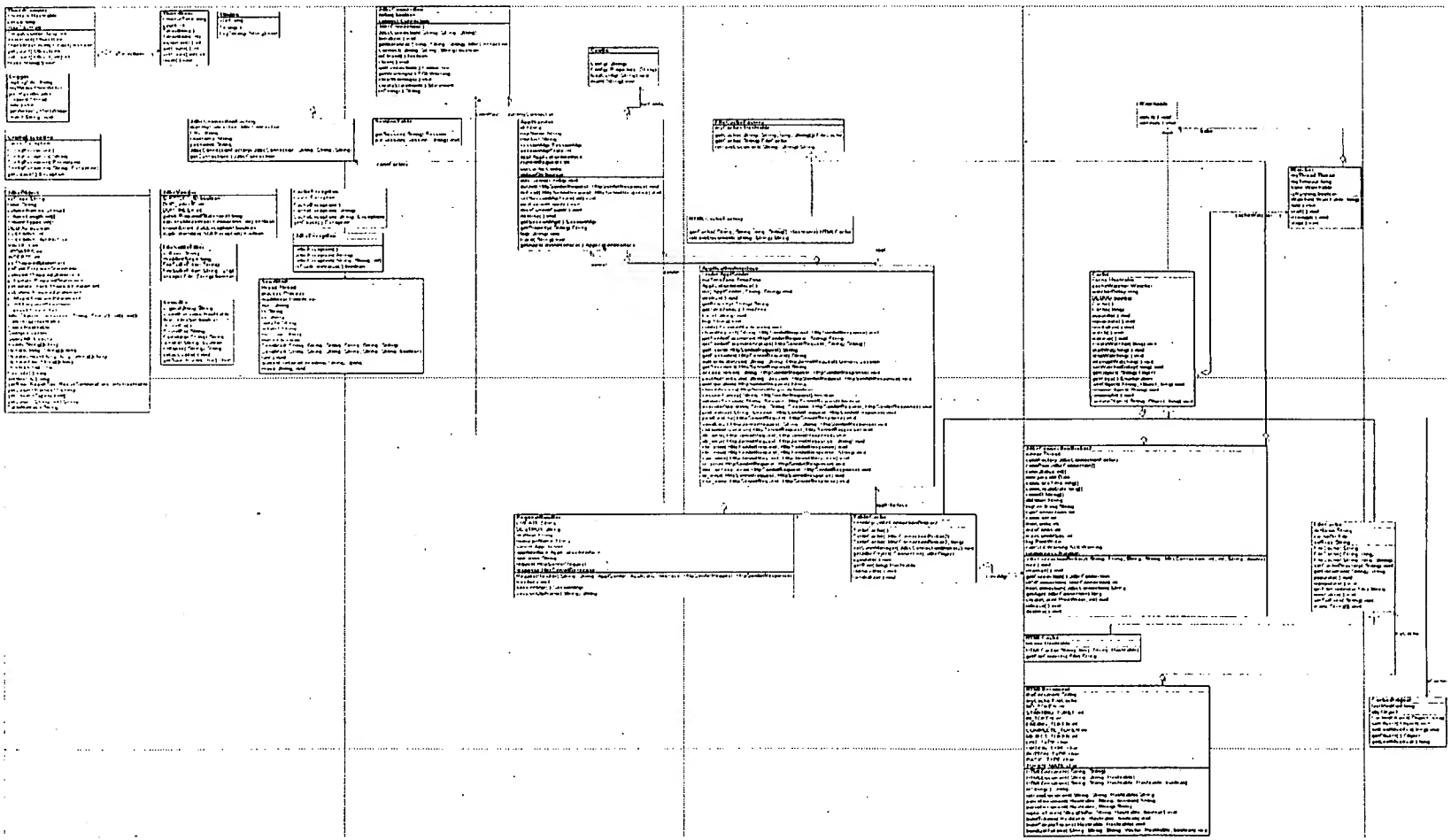


PersonalList

PersonalList
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psIsCardThere:PreparedStatement psJoinByUsersId:PreparedStatement psContainsCard:PreparedStatement psJoinByUsersIdName:PreparedStatement psDeleteByCardId:PreparedStatement
PersonalList(Connection) IsCardThere(long, long):boolean QueryJoinByUserId(long):Vector QueryContainsCard(long):Vector QueryJoinByUserIdName(long, char):Vector Insert(DatabaseConnection2, long, long, long, short):long DeleteByUserId(DatabaseConnection2, long):int DeleteByCardId(DatabaseConnection2, long):int Delete(DatabaseConnection2, long, long):int

PrivateList

PrivateList
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] psGetMask:PreparedStatement psUpdateMask:PreparedStatement psDeleteByCardId:PreparedStatement
PrivateList(Connection) Get(long, long):Hashtable UpdateMask(long, short):long DeleteByCardId(long):long



THIS PAGE BLANK (USPTO)

TimedCounter

TimedCounter
counters:Hashtable period:long maxCount:int
TimedCounter(long, int) increment(Object):int checkMaxCount(Object):boolean getCount(Object):int setCount(Object, int):int main(String[]):void

TimedItem

TimedItem
timeoutTime:long count:int
TimedItem( ) TimedItem(int) increment( ):int getCount( ):int setCount(int):int reset( ):void

Timing

Timing
start:long
Timing( ) LogTiming(String):void

Logger

Logger
myLogFile:String myWriter:PrintWriter pin:PipedReader
Logger(String) run( ):void getWriter( ):PrintWriter main(String):void

Config

Config
Config(String) Config(Properties, String) loadConfig(String):void main(String):void

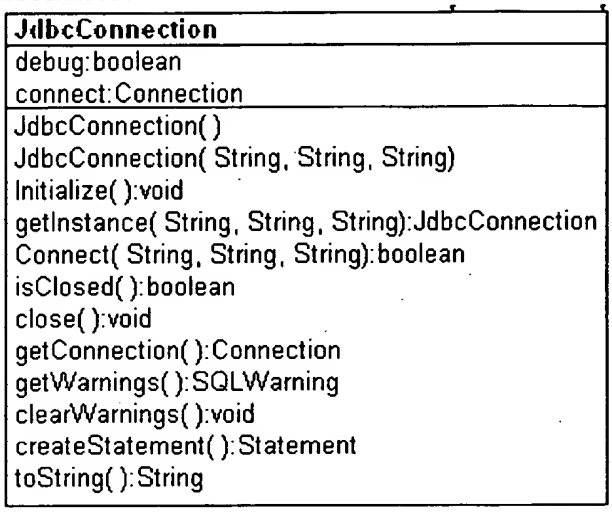
ConfigException

ConfigException
cause:Exception
ConfigException( ) ConfigException(String) ConfigException(Exception) ConfigException(String, Exception) getCause( ):Exception

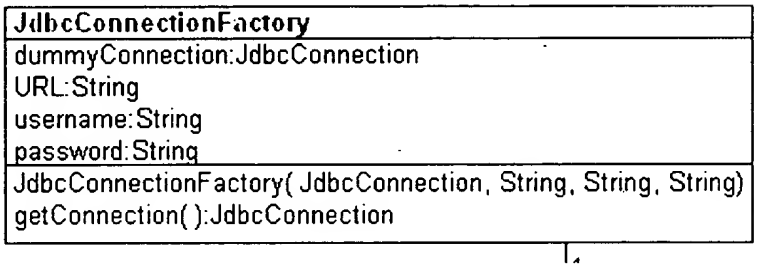
JdbcObject

JdbcObject
szClass:String table:String columnNames:String[] columnLength:int[] columnTypes:int[] DEBUG:boolean COLUMNS:int COLUMNS_INSERT:int VALUES:int UPDATES:int INSERTS:int ps:PreparedStatement psById:PreparedStatement psInsert:PreparedStatement psUpdate:PreparedStatement psUpdateCheck:PreparedStatement psDelete:PreparedStatement psMaxId:PreparedStatement psAll:PreparedStatement connect:Connection
JdbcObject(Connection, String, String[], int[], int[]) Get(long):Hashtable Get( ):Hashtable Query( ):Vector QueryAll( ):Vector Insert(String[]):long Update(long, String[]):long UpdateCheck(long, long, String[]):long UpdateRow(String[]):long Delete(long):long Execute( ):long getNextId( ):long getRow(ResultSet, ResultSetMetaData, int):Hashtable getColumnNames( ):String[] getColumnTypes( ):int[] getColumnString(int):String TableName( ):String

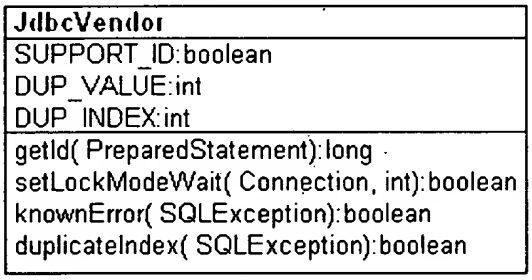
JdbcConnection



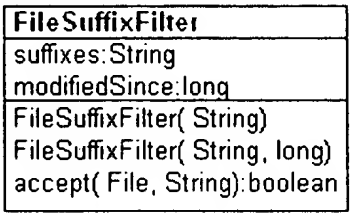
JdbcConnectionFactory



JdbcVendor



FileSuffixFilter



SoundEx

SoundEx
originalString:String soundExCodes:Hashtable AreCodesSet:boolean
SoundEx( ) SoundEx(String) Calculate(String):String isValid(String):boolean collapse(String):String setupCodes( ):void getSoundCode(char):short

CacheException

CacheException
cause:Exception
CacheException( ) CacheException(String) CacheException(String, Exception) getCause( ):Exception

JdbcException

JdbcException
JdbcException( ) JdbcException(String) JdbcException(String, String, int) isDuplicateValue( ):boolean

SendMail

SendMail
thread:Thread process:Process mailWriter:PrintWriter from:String to:String cc:String replyTo:String subject:String message:String mimed:boolean
SendMail(String, String, String, String, String, String) SendMail(String, String, String, String, String, String, boolean) run( ):void quotedPrintableEncoding(String):String main(String):void

SessionTable

SessionTable
getSession(String):Session putSession(Session, String):void

ApplServlet

ApplServlet
id:String mgrName:String rmiHost:String sessionMgr:SessionMgr sessionMgrState:int appl:ApplicationInterface currentRequests:int ourConfig:Config debugOn:boolean
init(ServletConfig):void doGet(HttpServletRequest, HttpServletResponse):void doPost(HttpServletRequest, HttpServletResponse):void setSessionMgrState(int):void incrCurrentCount( ):void decrCurrentCount( ):void destroy( ):void getSessionMgr( ):SessionMgr getProperty(String):String log(String):void trace(String):void getApplicationInterface( ):ApplicationInterface

FileCacheFactory

FileCacheFactory
myCaches:Hashtable
getCache(String, String, long, String[]):FileCache getCache(String):FileCache retrieveDocument(String, String):String

HTMLCacheFactory

HTMLCacheFactory
getCache(String, String, long, String[], Hashtable):HTMLCache retrieveDocument(String, String):String

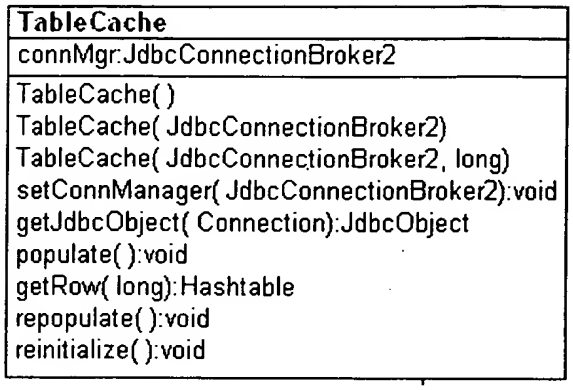
RequestHandler

RequestHandler
CREATE:String DESTROY:String rmiHost:String managerName:String servlet:ApplServlet applInterface:ApplicationInterface operation:String request:HttpServletRequest response:HttpServletResponse
RequestHandler(String, String, ApplServlet, ApplicationInterface, HttpServletRequest, HttpServletResponse) handle( ):void sessionMgr( ):SessionMgr sessionObjName(String):String

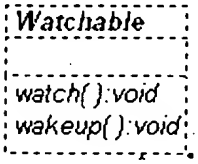
ApplicationInterface

ApplicationInterface
servlet:ApplServlet ourTimeZone:TimeZone
ApplicationInterface( ) init(ApplServlet, String, String):void destroy( ):void getProperty(String):String getTimeZone( ):TimeZone trace(String):void log(String):void notify(SessionNotification):void chainRequest(String, HttpServletRequest, HttpServletResponse):void getServletParameter(HttpServletRequest, String):String getServletParameterValues(HttpServletRequest, String):String[] getUserId(HttpServletRequest):String getPassword(HttpServletRequest):String authenticateUser(String, String, HttpServletRequest):GenericSession getSessionId(HttpServletRequest):String accessDenied(String, HttpServletRequest, HttpServletResponse):void postAuthenticate(String, Session, HttpServletRequest, HttpServletResponse):void getOperation(HttpServletRequest):String checkAccess(HttpServletRequest):boolean sessionFailure(String, HttpServletRequest):boolean validateSession(String, Session, HttpServletRequest):boolean executeOperation(String, String, Session, HttpServletRequest, HttpServletResponse):void preDestroy(String, Session, HttpServletRequest, HttpServletResponse):void postDestroy(HttpServletRequest, HttpServletResponse):void sendError(HttpServletRequest, String, String, HttpServletResponse):void unknownOperation(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse):void db_error(HttpServletRequest, HttpServletResponse, String):void nfe_error(HttpServletRequest, HttpServletResponse):void nfe_error(HttpServletRequest, HttpServletResponse, String):void sae_error(HttpServletRequest, HttpServletResponse):void re_error(HttpServletRequest, HttpServletResponse):void doc_access_error(HttpServletRequest, HttpServletResponse):void io_error(HttpServletRequest, HttpServletResponse):void nse_error(HttpServletRequest, HttpServletResponse):void

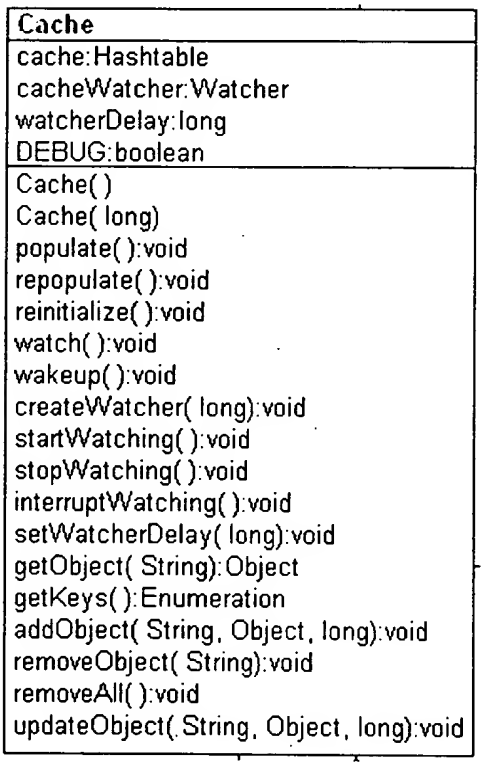
TableCache



Watchable



Cache



JdbcConnectionBroker2

JdbcConnectionBroker2
runner:Thread connFactory:JdbcConnectionFactory connPool:JdbcConnection[] connStatus:int[] now:java.util.Date connLockTime:long[] connCreateDate:long[] connID:String[] dbDriver:String logFileString:String currConnections:int connLast:int minConns:int maxConns:int maxConnMSec:int log:PrintWriter currSQLWarning:SQLWarning randomness:Random
JdbcConnectionBroker2(String, String, String, String, JdbcConnection, int, int, String, double) run( ):void interrupt( ):void getConnection( ):JdbcConnection idOfConnection(JdbcConnection):int freeConnection(JdbcConnection):String getAge(JdbcConnection):long createConn(PrintWriter, int):void release( ):void destroy( ):void

HTMLCache

HTMLCache
tokens:Hashtable
HTMLCache(String, long, String, Hashtable) getFileContents(File):String

HTMLDocument

HTMLDocument
theDocument:String myCache:FileCache NO_TOKEN:int STARTING_TOKEN:int IN_TOKEN:int ENDING_TOKEN:int COMPLETE_TOKEN:int REJECT_TOKEN:int LIST_TYPE:char OPTION_TYPE:char BUTTON_TYPE:char BASIC_TYPE:char TOKEN_MATE:char
HTMLDocument(String, String) HTMLDocument(String, String, Hashtable) HTMLDocument(String, String, Hashtable, Hashtable, boolean) toString( ):String retrieveDocument(String, String, Hashtable):String parseDocument(Hashtable, String, boolean):String parseDocument(Hashtable, String):String replaceToken(StringBuffer, String, Hashtable, boolean):void buildTokens(Hashtable, Hashtable, boolean):void buildSimpleTokens(Hashtable, Hashtable):void buildListTokens(String, String, String, Vector, Hashtable, boolean):void

Watcher

Watcher
myThread:Thread myTimeout:long babe:Watchable isRunning:boolean
Watcher(Watchable, long) run( ):void start( ):void interrupt( ):void stop( ):void

FileCache

FileCache
dirName:String cacheDir:File suffixes:String
FileCache(String) FileCache(String, long) FileCache(String, long, String) setCacheDirectory(String):void getDocument(String):String populate( ):void repopulate( ):void getFileContents(File):String reinitialize( ):void setSuffixes(String):void main(String[]):void

CacheObject

CachedObject
lastModified:long obj:Object
CachedObject(Object, long) setObject(Object):void setLastModified(long):void getObject( ):Object getLastModified( ):long